UCBlocker: Unwanted Call Blocking Using Anonymous Authentication

Changlai Du and Hexuan Yu, *Virginia Tech;* Yang Xiao, *University of Kentucky;*
Y. Thomas Hou, *Virginia Tech;* Angelos D. Keromytis, *Georgia Institute of Technology;*
Wenjing Lou, *Virginia Tech*

https://www.usenix.org/conference/usenixsecurity23/presentation/du

This paper is included in the Proceedings of the
32nd USENIX Security Symposium.

August 9–11, 2023 • Anaheim, CA, USA

978-1-939133-37-3

UCBlocker: Unwanted Call Blocking Using Anonymous Authentication

Changlai Du
Virginia Tech
cdu@vt.edu

Hexuan Yu
Virginia Tech
hexuanyu@vt.edu

Yang Xiao
University of Kentucky
xiaoy@uky.edu

Y. Thomas Hou
Virginia Tech
thou@vt.edu

Angelos D. Keromytis
Georgia Institute of Technology
angelos@gatech.edu

Wenjing Lou
Virginia Tech
wjlou@vt.edu

Abstract

Telephone users are receiving more and more unwanted calls including spam and scam calls because of the transfer-without-verification nature of global telephone networks, which allows anyone to call any other numbers. To avoid unwanted calls, telephone users often ignore or block all incoming calls from unknown numbers, resulting in the missing of legitimate calls from new callers. This paper takes an end-to-end perspective to present a solution to block unwanted calls while allowing users to define the policies of acceptable calls. The proposed solution involves a new infrastructure based on anonymous credentials, which enables anonymous caller authentication and policy definition. Our design decouples caller authentication and call session initiation and introduces a verification code to interface and bind the two processes. This design minimizes changes to telephone networks, reduces latency to call initiation, and eliminates the need for a call-time data channel. A prototype of the system is implemented to evaluate its feasibility.

1 Introduction

Telephony systems, which enable individuals to place phone calls to any phone number, have seen significant success over the past century. At that time, people used to answer the phone when it rang, and expect that their calls to others would also be answered. However, phone calling has declined in popularity in recent years. People are less reliant on phone calls, and they are stopping answering calls from unknown numbers. The expectation of pickup as a core aspect of the telephone culture is disappearing. This trend is evidenced by the polls conducted by the New York Times for the 2018 Midterm Elections [1], in which over 2.82 million phone calls were made across the US to the callees who didn't know the call was from NYT for a poll. Only 1.71% of the calls were answered.

There are several reasons why people are not willing to pick up unknown calls. One reason is that people have become accustomed to communicating through alternative methods, such as messaging apps and social media, which are perceived as less intrusive and do not require an immediate response. Another key reason is the increasing prevalence of spam and scam calls. Similar to email spam, *spam calls* are a type of unwanted calls made to a large number of people at once. *Scam calls*, on the other hand, are calls with the intention of tricking or defrauding the callee in order to obtain personal or financial information. According to a report by Truecaller [2], Americans received an average of 20 spam calls per month in 2022, which amounts to approximately 8 billion spam calls per month, and an estimated $39.5 billion was lost to phone scams in 2022 in the US.

Two main technical factors have contributed to the rampancy of spam and scam calls—the growth of Voice over Internet Protocol (VoIP) usage and caller ID spoofing. *VoIP technology*, which gained widespread popularity in the mid-2000s, allows spammers to use auto dialers (often referred to as *robocalls*) to make mass spam calls at low costs, as cheap as $0.01 per minute [3]. Additionally, VoIP enables individuals to evade prosecution for spamming and scamming by making calls across jurisdictional lines. *Caller ID spoofing* involves altering the caller ID information (phone number and/or name) in signaling messages, allowing the caller to impersonate others, like government agencies, businesses, the callee's neighbors, or even the callee's contacts. According to a study by Tu et al. [4], users are more likely to pick up on spoofed calls if the caller ID has been carefully altered to appear legitimate.

There have been various attempts by governments, businesses, and academic communities to address the problem of spam and scam calls [5–19]. However, these efforts have their own limitations. Some need to trade away usability considerably by disturbing the callee too much, introducing significant delays, or disabling legitimate incoming calls from unknown numbers altogether. Some are hard to deploy due to the required significant upgrades to the network infrastructure. Others are not robust because of caller ID spoofing attacks. Readers are referred to [3] for a comprehensive survey on these solutions. Despite these efforts, the volume of spam calls remains high and has not declined [2], indicating that these solutions are not sufficient to effectively solve the problem.

In this paper, we present UCBlocker, an end-to-end unwanted call blocking system that makes minimal changes to the existing telephone networks and calling process. UCBlocker allows the *callee* to authenticate the caller based on the caller's *attributes*. We use anonymous credentials (ACs) for caller authentication. Before making a phone call, the caller is required to provide proof of certain attributes in her identity which satisfy the callee's pre-defined policies. The callee will issue a verification code to the caller if the authentication is successfully completed, which is inspired by two-factor authentication used by many web services. UCBlocker binds the authentication over data channel to the call session in telephone networks using this verification code. We explore possible methods of transmitting the code through telephone networks and find that re-purposing the caller ID for carrying the code poses a novel and efficient option. The callee checks the verification code to determine if the call is pre-authenticated.

Compared to previous designs, UCBlocker has several advantages in usability, deployability and robustness. First, it is user-centric and user-controlled. The callee defines what (rather than which) phone calls are wanted using attribute-based policies (rather than the caller ID), which enables incoming calls from legitimate unknown numbers. Second, the end-to-end caller authorization can be implemented as a separate application and Internet service, complementing the calling function, which requires minimal changes to the telephone networks. Third, the process of caller authentication (to acquire the verification code) and call session setup can be decoupled, which introduces minimal delays to call initiation.

We make the following contributions:

- We design a new unwanted call blocking architecture named UCBlocker, which allows the callee to set up authentication policies and anonymously authenticate the caller using attributes of caller's identities based on anonymous credentials. Our design enables legitimate unknown incoming calls, requires minimal changes to the telephone networks, and introduces negligible delays to the call setup process.

- We explore the options of binding the anonymous authentication results with call sessions and evaluate their feasibility in telephone networks. We find that re-purposing the caller ID header field in signaling messages to transmit the verification code is promising in that it does not require any change to the signaling protocols in the telephone networks, which relieves the network operators of caller ID attestation.

- We implement a prototype of UCBlocker and evaluate the feasibility and performance of the system. We measure the time cost of caller authentication. The result show that code verification delay is negligible. For consecutive calls after the authentication, the additional latency of the authentication process is about 1.5s.

With UCBlocker, we aim to address the issue of unwanted calls from the perspective of the callee. Functionally, our design will: 1) allow calls from the callee's contacts with minimal disruption; 2) allow unknown but callee-defined welcome calls to reach the callee; and 3) effectively block all unwanted calls out of the callee's preferences. Considering the numerous advantages for consumers including enhanced fraud protection, reduced disturbance and frustration, and less wasted time, as well as benefits to businesses and governments such as increasing the likelihood of customer engagement and enhancing trust, we hope UCBlocker will rebuild trust in telephone networks and revitalize the use of phone calls.

The rest of this paper is organized as follows. Section 2 provides background information about the telephone networks and anonymous credentials. Section 3 describes the overview of our design, including the system model, adversary assumptions, design requirements, and some use cases. Section 4 formally specify the design of the authentication protocols based on anonymous credentials. Section 5 describes the binding of authentication and call session as well as its security discussion. Section 6 provides details of our implementation and the results of our evaluation experiments. Section 7 makes some clarification discussion. Section 8 compares our work with related work and Section 9 concludes the paper.

2 Background

This section provides a brief introduction to the telephone network architecture and the anonymous credential concepts.

2.1 Modern Telephone Networks

The global telephone network is a system that connects various types of telecommunication networks, including Public Switched Telephone Networks (PSTN), mobile cellular networks, and Voice over Internet Protocol (VoIP) networks. These networks are connected through gateways located at their boundaries. In a VoIP network, IP devices are connected to a Private Branch Exchange (PBX), which can routes phone calls over IP networks. In order to make phone calls to PSTN landlines and mobile phones on cellular networks, the PBX must be connected to a VoIP service provider that offers trunking services though an IP/PSTN gateway.

In order to initiate and terminate a phone call, a control channel is used for signaling and an end-to-end voice channel is used to transmit the voice data. There are various protocols that are used for signaling, including Signaling System 7 (SS7) [20], Session Initiation Protocol (SIP) [21], and

H.323 [22]. Telephone service providers provide caller ID services, which transmit the caller's phone number and/or name to the recipient so that they can identify the caller before answering the call. In traditional PSTN and cellular networks, the caller ID is generated and/or authenticated by the network. However, in VoIP networks, the caller ID is generated on the caller's side which can be set to any value. Some VoIP service providers even offer call ID spoofing as a featured service. The low cost and capability to set arbitrary caller IDs make VoIP networks a common location to start telephone spam and scams.

2.2 Anonymous Credentials

Attribute-based authentication. Anonymous Credential systems [23, 24] allow users to prove their identities satisfying certain properties without revealing the identity details. This type of authentication provides increased privacy, as only the necessary attributes are leaked, rather than the user's full identity. In advanced systems, predicates can also be tested, further reducing the amount of information leaked.

Roles. An anonymous credential system has three key players: the holder, issuer, and verifier. The holder holds one or more credentials that correspond to their different identities. The issuer creates and issues these credentials to the holder, making assertions about their attributes. When the holder wants to access a resource, they present their credentials to the verifier anonymously. The verifier then checks the presentations to determine if access should be granted.

Features. Anonymous credentials have four key features that set them apart [25]. *Selective Disclosure* allows the credential holder to select which attributes to reveal to the verifier, while keeping the rest private. *Unlinkability* randomizes the issuer's signature, preventing it from serving as a correlating factor. *Private Holder Binding* binds the credential to the holder without creating any information that needs to be revealed during presentation. *Predicates* allow hidden values to be used in operations with values provided by the verifier, such as proving that a bank account balance is above a certain threshold without revealing the exact balance.

W3C DID scheme. W3C has published standards for Decentralized identifiers (DIDs) [26] and Verifiable Credentials [27]. DIDs are a type of identifiers that enable verifiable and decentralized digital identity. DIDs are generated by the identifier owners. A DID can be resolved to a DID document containing the public keys of the DID owner, which is stored in a verifiable data registry like a distributed ledger. W3C also defines the Verifiable Credential data model, which is a set of claims about the holder that can cryptographically verified. Anonymous credentials, which provide anti-correlation properties through technologies like zero-knowledge proofs, are a type of verifiable credential.

The anonymous credential system is the main cryptography building block of the design for UCBlocker, and W3C DID scheme facilitates the discovery of the anonymous credential services from telephone numbers.

3 UCBlocker Design

Telephone networks are becoming increasingly complex, especially with their convergence with the Internet. We will omit the network details and use a simplified system model to characterize UCBlocker as an end-to-end solution. We present the system model, security assumptions, and our design overview in this section.

3.1 System Model

UCBlocker consists of two subsystems: the distributed anonymous authentication function over the Internet and the call session initiation and verification function over the telephone networks. The same entity has different roles in the two subsystems. Without loss of generality, we refer to Alice as the **caller/ credential holder**, and Bob as the **callee/verifier**. We also refer Alice and Bob to their devices or the UCBlocker applications installed on their devices (known as the user agents). The devices must have some level of processing power in order to support the installation of UCBlocker. Examples of such devices include smartphones, VoIP phones, computers, or smart devices connected to landline PSTN phones with internet access. Alice and Bob have UCBlocker installed on their devices and use it to initiate and answer phone calls.

In addition to Alice and Bob, there are other participants involved in the distributed anonymous authentication system. The **issuer** is responsible for issuing credentials to Alice, which can be individuals who are direct contacts of Alice, or organizations who have connections with Alice in real life as defined in W3C [27]. In an ideal scenario where digital identity and anonymous credentials are ubiquitously deployed, an issuer refers to any organization or entity that holds the responsibility of issuing specific documents or credentials within a particular context, and its identity can be publicly verified. For example, the Department of Motor Vehicles (DMV) serves as an issuer. Notably, digital driver's licenses have already been introduced in several US states, including Arizona, Colorado, Maryland, and Georgia, and more states are also committed to support the feature [28]. To facilitate the initial implementation of UCBlocker, a Mobile Virtual Network Operator (MVNO) can act as an issuer to verify and assert the attributes of UCBlocker users before the widespread deployment of the AC infrastructure. This approach presents a valuable business opportunity for an MVNO to attract security-conscious customers who value UCBlocker's capabilities. Alternatively, UCBlocker is designed to allow third parties to assume independent roles, such as that of an issuer or verifier, with the possibility of receiving compensation from subscribers. The public storage, implemented as a **public ledger**, provides secure and trusted record-keeping and querying services. Bob's verifier is implemented as a *service endpoint*, which should be always available for potential callers.

3.2 Adversary Model

We consider an adversary (Eve) who does not hold any valid credentials that Bob accepts. As an attacker, Eve's goal is to make a successful phone call to Bob. In our call blocking context, a phone call is successful as long as it reaches Bob, regardless Bob answers the call or not. Eve may or may not be a UCBlocker user. Eve may perform several attacks:

Attacks to authentication. Eve may intercept the communication between the parties involved to acquire either the anonymous credential during its issuance or the credential presentation during its transmission. Eve may try to forge the credential or replay the presentation to deceive the verifier to issue her a verification code. Eve may eavesdrop the verification code when it is distributed from the callee to the caller and place a phone call using the code *before* Alice, which is named *code eavesdropping attack*.

Attacks to call initiation. Eve may eavesdrop the verification code when it is transmitted in the telephone system to initiate the call. Eve may try to use the code to place another call. If Eve places the call *concurrently* with Alice, we name it *race attack*. Otherwise if Eve places the call at a later time, we name it *code replay attack*.

The design of UCBlocker is based on the following security assumptions. We assume that Eve is unable to break the cryptography used in the anonymous credentials. Even if she gets the credential, she can not construct valid presentations without the holder's private key. We also assume that both the telephone and Internet service providers are trusted and will follow their service protocols. The verification code works as a one-time password so it is not susceptible to code replay attacks, and we will discuss the code race attack in Section 5.4. UCBlocker deals with the presentation replay attack and the code eavesdropping attack in the design of the protocols.

3.3 Requirements

Given the adversary assumptions outlined above, UCBlocker aims to achieve the following design goals.

Security. The unwanted call blocking system should guarantee that only calls that follow the callee's policies can get through to the callee. An adversary cannot bypass the verification process to make disturbance to the callee.

Usability. The call blocking system should enable legitimate calls from unknown numbers as long as the caller can provide proof of holding accepted credentials. Additionally, the system should be user-friendly and require minimal extra effort from both the caller and the callee. For the callee, this means that their only responsibilities would be creating policies and making any necessary updates. The caller may need to interact with the call blocking system to select the required attributes and authorize the disclosure of these attributes to the callee for verification to obtain verification codes.

Compatibility. The call blocking system in UCBlocker should make minimal changes to the telephone networks. Modifying telephone networks can be both costly and time-consuming.

Efficiency. The call blocking system should have minimal computation and communication costs, so it does not add significant delays to the original call session setup and can be deployed on devices with limited resources.

3.4 Use Cases

There is a wide range of use cases of UCBlocker. We highlight some common examples and explain how to use anonymous credentials in their implementation.

Bob accepts calls from his contacts only. In this simplest use case, Bob can issue credentials to his contacts, indicating their legitimacy to call him. In fact, this can also be achieved by using a local white list. However, setting a policy to block all unknown calls is not a satisfying solution, as most telephone users still want to be reachable by people they have not spoken with before.

Bob gives his phone number to others and expects their callbacks. Examples of this use case include scenarios where Bob applies for a job online or makes an appointment with a specialist doctor. In these cases, Bob can issue a credential and send it to the potential caller along with his phone number. Additionally, Bob can add more attributes to the credential, such as setting an expiration date or a maximum number of times the credential can be used to call him.

Bob accepts calls from those who have authority-issued credentials. This can include situations where an employee from Bob's bank contacts him about suspicious activity on his account, or a local plumber advertises his services to Bob, where the plumber can prove his location by sharing the zip code of his home address from his driver's license issued by DMV. Bob can set his policy to only accept calls from people in his immediate area by creating a list of approved zip codes. This way Bob can accept calls from his *real* neighbors, rather than being bothered by *neighbor spoofing* scammers.

Bob sets a policy requiring multiple credentials. We can use the following scenario as an example. Bob is searching for a local, licensed plumber and has advertised his phone number on a local billboard. In order for a plumber to contact Bob, he must provide evidence that he is in the area (by showing the zip code on his driver's license) and that he possesses a valid plumber certificate from a governing authority.

The `School` case study. Consider a scenario in which an elementary school teacher, Alice, is attempting to contact a parent, Bob, for urgent notice. Alice is not in Bob's contact list, but she has a credential issued by the School to prove her identity. Bob has set his policy to accept phone calls from teachers of the School. This scenario will be used as a running example to demonstrate the core designs of the UCBlocker system in this paper.

MVNO as the bootstrapping issuer. Prior to the wide availability of the AC infrastructure, it is possible to establish an MVNO that can serve as an issuer of credentials for UCBlocker users. In the specific case of the School scenario, the MVNO can undertake the verification process of Alice's work certificate and subsequently issue a credential that attests to her employment at the school, particularly if the school itself is unable to provide such a credential at that time.

3.5 Design Overview

Blocking unwanted calls ultimately requires end-to-end caller authentication, because the recipient is the only one who can determine which calls are welcome. There are various considerations when designing such an authentication scheme.

Authentication channel. Telephone networks typically use two channels for creating a phone call: a *control channel* for signaling and an end-to-end *voice channel* for voice data transmission. It is currently infeasible to achieve end-to-end authentication using the control channel because different telephone networks use different signaling protocols. All the protocols and devices will need to be updated, otherwise, the authentication information carried by signaling messages may be lost when they are translated at the gateways connecting the telephone networks. On the other hand, it is possible to used the voice channel for end-to-end authentication, as exampled by Reaves et al. in AuthLoop [12]. However, AuthLoop introduced additional delays because implementing security protocols (e.g. TLS) on the narrow band voice channel is time inefficient. The third option is the *data channel*. With the convergence of telephone networks to IP networks, more telephone devices have access to a data channel, which connects the devices to the Internet. UCBlocker takes advantage of the data channel for caller authentication.

Centralized or distributed. End-to-end authentication can be implemented either centralized or distributed. However, because of the unique requirement of unknown caller authentication in telephone networks, a distributed solution is more feasible. An unknown caller must reveal some information of his/her identity to the callee to get authenticated, where an identity is defined as a set of claims made by one subject about itself or another subject [29]. The claims are attributes of the caller, which are usually privacy-sensitive information. In a centralized architecture, the central server is responsible for collecting and forwarding these attributes, which will pose a significant risk to the callers' privacy. UCBlocker adopts the distributed approach and employs anonymous credentials to safeguard privacy during authentication.

Binding authentication to call session. Once authentication is completed over the data channel, the next question is how to link it to a call session. One option is to rely on a trusted third party to help with call initiation as shown by Reaves et al. in Authenticall [13] . An alternative is to generate a one-time-use shared secret after authentication, which can be sent from the caller to the callee over telephone networks when the call session is established. In this paper, we refer to the authentication secret as a *verification code*, similar to the code used in two-factor authentication systems on the Internet. Voice channel can be used to transmit the verification code by using a data-voice modem. Control channel is another option, but transmitting the secret using control channel can be tricky because of the message translation at the network gateways. One of the few pieces of information that can transit between networks is the caller ID. Because of its declarative nature, it is possible to use caller ID to carry the authentication secret. We call it caller ID *customization* instead of caller ID *spoofing* because our intent is not to spoof other phone numbers but to transfer the authentication secrets. Another benefit of using the verification code is that its generation and usage can be decoupled. The caller can acquire multiple codes during one authentication, store them in his device and use them later. Using caller ID to send the code and start a phone call introduces no extra delays to the call session setup process.

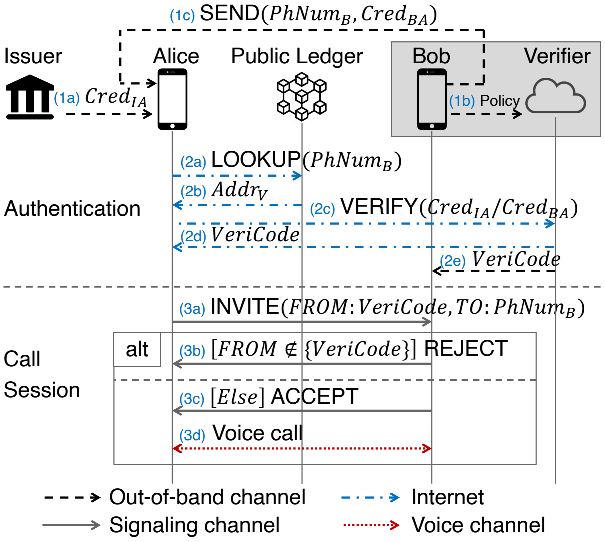


Figure 1: UCBlocker overview.

In summary, UCBlocker uses anonymous credentials for caller authentication and explore different methods of carrying the verification code for channel binding. The overview of UCBlocker is depicted in Figure 1. There are five main participants, including the caller Alice (also the credential holder), the callee Bob, the credential Issuer, a Public Ledger and the Verifier which is a service endpoint working as a verification agent for Bob.

The workflow of UCBlocker is broken down into three phases: the setup phase, the authentication phase, and the call initiation phase. In the setup phase, Alice requests anonymous credentials from Issuer (1a and 1c), Bob sets his policies encoding his preferences of wanted calls (1b), and Alice obtains the contact information of Bob (1c). Message 1c is an example where Bob sends his phone number to Alice, together with a credential proving Alice has been added to his contacts. In cases where Alice is an unknown caller, she can also get Bob's phone number from any other sources. In the authentication phase, Alice looks up Bob's service endpoint on the public ledger (2a, 2b) and presents her proof to Bob to get the verification code (2c-2e). Later when Alice wants to make a phone call to Bob, she initiates the call by sending the verification code to Bob (3a, sending the code via caller ID header field is depicted), which Bob can verify to accept or block the call request (3b-3d).

3.6 Deployment

The UCBlocker client is designed as an application. For the callee, UCBlocker assists users in selecting attributes and configuring policies. In cases where incoming calls fail to meet the specified policies, UCBlocker silently blocks these calls. Users also have the option to redirect blocked calls to their voicemail. Detailed discussions regarding policy deployment can be found in Section 4.1.2. On the caller's side, it is necessary to initiate phone calls using the UCBlocker application. If there are no available verification codes, UCBlocker will authenticate the caller to the callee. During this authentication process, UCBlocker automatically matches the callee's policies against the caller's credentials. The caller will receive a warning when certain attributes need to be revealed for successful authentication. To provide a visual representation of how callers can choose to expose specific attributes when making a phone call, a user interface example is included in the Appendix A.

In order to facilitate the initial implementation of our scheme, UCBlocker incorporates a public ledger. This ledger serves as a secure and trusted public database, storing crucial public information within the system. This includes the published schemas of issuers, the public keys of all stakeholders, and the decentralized identifiers/documents associated with UCBlocker users. By maintaining this public ledger, UCBlocker ensures the availability and integrity of essential data, contributing to the overall functionality and security of the system.

Before the credential issuance infrastructure is fully functional, a practical solution to start UCBlocker service is to set up an MVNO that serves as the bootstrapping issuer. UCBlocker users can present their proofs of certain attributes (e.g. name/home address on a physical driver's license) to the MVNO to get a credential issued by the MVNO. However, as the credential issuance infrastructure becomes more widely deployed, users are expected to acquire credentials from other "formal" issuers. For example, obtaining a digital driver's license from the DMV can replace the credentials initially issued by the MVNO. This transition promotes the use of credentials from established and recognized issuers, enhancing the credibility and reliability of the UCBlocker system.

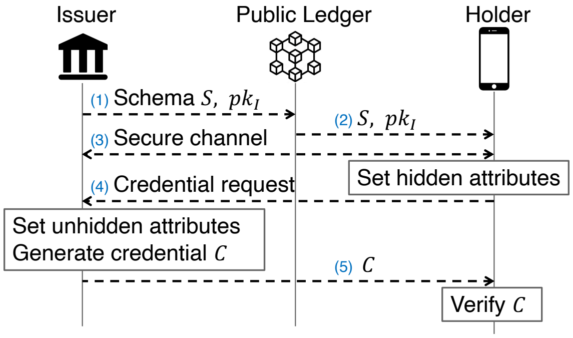


Figure 2: Credential issuance protocol.

4 Authentication

This section provides a thorough explanation of attribute-based anonymous authentication protocols, including the steps involved in both the setup and authentication phases.

4.1 Setup Phase

The setup phase encompasses three types of activities: the caller obtaining credentials (1a, 1c in Figure 1), callee deploying policies (1b), and caller acquiring callee's contact information (1c). These activities are typically conducted through out-of-band channels, and there are usually multiple options available. For instance, Alice may obtain her employment credential by logging in the School website or visiting the HR office, and Bob may provide his phone number to Alice through social media or during a face-to-face meeting.

4.1.1 Credential Issuance

An anonymous credential is a type of attribute-based credential that enables users to prove certain attributes without revealing additional information or being linked across multiple credential presentations.

The credential issuance protocol is illustrated in Figure. 2. In message 1, issuer defines a credential schema S, selects the public key pk_I to be used with this schema, and publishes S and pk_I on the ledger. A credential schema is the data template for a credential [30], which defines the list of attributes included in the credential. It serves as a shared point of trust for the issuer, holder, and verifier. For each attribute, its name, data type and whether it's hidden from issuer are specified. If an attribute is hidden from issuer, its value will be set by holder and blinded before sent to issuer for signing(i.e. blind signature). E-voting system is a typical application scenario of blind signature, where the local authority checks and signs the eligibility of a voter, without learning the voter's choice. As an example of credential schema, in the **School** case, the School may define a two-attribute schema for its employment credentials simplified as `{"name":{"type":string,"hidden":false}," employed":{"type":boolean,"hidden":false}}`. If the MVNO works as the issuer instead of the School, one more attribute is needed: `"school":\{"type":string,"hidden ":false\}`.

The form of pk_I varies based on the chosen signature scheme. In this paper, we use BBS+ signature [31]as an example to present the cryptography processes. Other signature schemes are available in the literature, and UCBlocker has cryptographic portability. Let $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ be groups of prime order p. Issuer sets up a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Assuming there are L attributes in the credential, issuer randomly takes generators $(h_0, h_1, \dots, h_L) \leftarrow \mathbb{G}_1^{L+1}$, $g_1 \leftarrow \mathbb{G}_1$, $g_2 \leftarrow \mathbb{G}_2$, and integer $x \leftarrow \mathbb{Z}_p^*$. Issuer then computes $w = g_2^x$, and sets $pk_I = (g_1, g_2, w, h_0, h_1, \dots, h_L)$. The private key sk_I is x.

In message 2, the holder retrieves S and pk_I from the public ledger. Step 3 is a mutual authentication between issuer and holder to set up a secure channel. Holder then assigns values to the hidden attributes according to the credential schema and sends a *credential request* to issuer in message 4 which includes the hidden attribute values. Without loss of generality, we assume that the attributes of the credential are encoded into integers $(m_1, \dots, m_L) \in \mathbb{Z}_p^L$. For an attribute m_i whose value is hidden to issuer, the blinded value will be $M_i = h_i^{m_i}$. We reserve m_1 for a secret sk_H which is chosen by holder as an unique identifier. The secret is known to holder only and is hidden before it is sent to issuer. The secret is used to combine multiple attributes from different credentials to form a single proof. It is named *link secret* by the Hyperledger [32] community as it literally links multiple credentials to the same holder.

Issuer will proceed by setting the unhidden attributes and generating the credential C. In the **School** case, School/MVNO assigns the attributes `{name="Alice", employed=true}` for Alice. The attributes will be encoded to integers, but for simplicity, we just denote `{m_2="Alice", m_3=true}` without causing ambiguity. Issuer selects random numbers $e, s \leftarrow \mathbb{Z}_p$ and computes

$$B = g_1 h_0^s \prod_{i=1}^{L} h_i^{m_i}, \quad A = B^{\frac{1}{e+x}} \qquad (1)$$

Issuer returns the credential $C = (A, e, s, \mathcal{M}_I)$ to holder, where $\mathcal{M}_I$ is a set consisting of the values assigned by issuer to the unhidden attributes.

Holder verifies C by checking

$$e(A, w g_2^e) \stackrel{?}{=} e(B, g_2) \qquad (2)$$

Holder will add the hidden attributes to C and store it as $C = (A, B, e, s, \{m_i\}_{i \in [1,\dots,L]})$.

4.1.2 Policy Deployment

UCBlocker is an authentication system that allows each user to set their own unique authentication policies for accepted calls. The policies must be easily understandable by non-technical users, and must be clearly defined so they can be translated into credential presentation request messages.

We define a universal set of public attributes $\mathcal{U}$ for telephone users in UCBlocker. Each credential contains a subset of attributes $\mathcal{A}_C \in \mathcal{U}$ and an issuer can create multiple credential schemas. To create a policy, Bob can either search for an issuer and then select the desired credential and attributes, or search for specific attributes and then choose from the available credentials which contains the attributes. As mentioned earlier, the credential schemas are stored on the public ledger. The UCBlocker client provides assistance to facilitate the search process and display the results to Bob. Bob then selects the desired attributes and incorporates predicate logic to define the policies. For detailed information on policy creation and formulation and a user interface example, please refer to Appendix A.

UCBlocker translates policies into a credential *presentation request*, which specifies what attributes from which credentials are accepted. A presentation request is a disjunction of multiple policies. Each policy is a set of attributes combined using predicate logic [33]. In the **School** case, a presentation request for Bob's policy looks like

```
{"policies":[{{
    "employed":True,
    "schema":"did:schema:abcde"}}]}
```

In this example, there is one policy defined in the presentation request, which contains an *equal* predicate to check if the value of `employed` attribute is `True`. `schema` defines the universal unique identifier of the schema used. We will discuss the zero-knowledge proof of predicate logics in Section 4.2.2.

4.1.3 Contact Credentials

Besides credentials issued by third parties, callees can issue their own *contact credentials*. UCBlocker defines a contact credential schema containing three attributes `{"name"," phone number","contact"}`. Using this, UCBlocker helps users to issue credentials to their contacts.

If Alice is already in Bob's contact list, Bob issues the credential $Cred_{BA}$ and sends it to Alice through any communication channel. If Bob and Alice are new friends and Bob is adding Alice to his contacts, Bob shares his phone number and issues the credential to Alice at the same time. Otherwise if Alice is an unknown caller, she can get Bob's phone number from any sources (e.g. Bob's personal website, yellow pages, their mutual friends), but she will resort to third party credentials to get authenticated by Bob.

4.2 Authentication phase

The authentication phase includes the processes of callee's verification service endpoint (the verifier) lookup (2a in Figure 1), credential presentation (2c), and verification code generation and distribution (2d, 2e).

4.2.1 Verifier Lookup

UCBlocker utilizes a public ledger to facilitate the retrieval of verifier's address. We adhere to W3C standards [26], utilizing Decentralized Identifiers (DID) and DID documents to store information related to those identifiers.

Decentralized identifiers. As previously stated in Section 4.1.1, each user selects a secret key sk to prove ownership of a specific credential and to link multiple credentials to the same holder. The secret key identifies the user cryptographically and must be masked before being transmitted to issuer or verifier to ensure unlinkability. A cryptographic tool is commitment schemes such as the Pedersen Commitment [34]. The public parameters of Pedersen Commitment consist of a group $\mathbb{G}$ of prime order p and generators $(g_0, \cdots, g_m)$. To commit values $(v_0, \cdots, v_m) \in \mathbb{Z}_p^m$, the commitment is calculated as $C = g_0^r \prod_{i=1}^m g_i^{v_i}$, where $r \in \mathbb{Z}_p$ is a random value. Using Pedersen Commitments, an arbitrary number of public keys can be generated from the same private key sk by choosing different random numbers r, resulting in $pk = g^{sk} h^r$. These generated public keys serve as pseudonyms for the user in the pairwise relationship between holder and issuer or verifier.

A DID is a globally unique identifier generated by the *user* with four essential characteristics: decentralized, persistent, cryptographically verifiable, and resolvable. DID is defined by W3C in the format of `scheme:method:method-specific-identifier`. For example, `did:example:123456789abcdefghi`. For cryptographically verifiability, DID should be generated from the user's keys, e.g. $DID = H_m(pk)$, where H_m is a hash function specific to DID methods. A DID resolves to a *DID document* stored on the public ledger. DID document typically contains the DID, verification methods including the public keys, and a set *service endpoints*. A service endpoint is a network address at which to communicate or interact with the DID owner.

UCBlocker features a digital wallet component that manages DIDs and credentials. The wallet generates a public DID using the user's secret key, along with the related DID document, and stores it on the public ledger to register the user's public DID.

Verifier lookup with a phone number. Phone number is more human-friendly than DIDs. In UCBlocker context, it is desirable to discover the verifier address starting from a phone number. To map the phone number to a DID and the verifier address in a verified and trusted way, we store the callee's phone number in his DID document. When a DID document is registered on a public ledger, the ledger will send a verification code to the phone number to prove its ownership. A caller looks up the callee's DID using his phone number, which resolves to a DID document on the public ledger containing the verifier address.

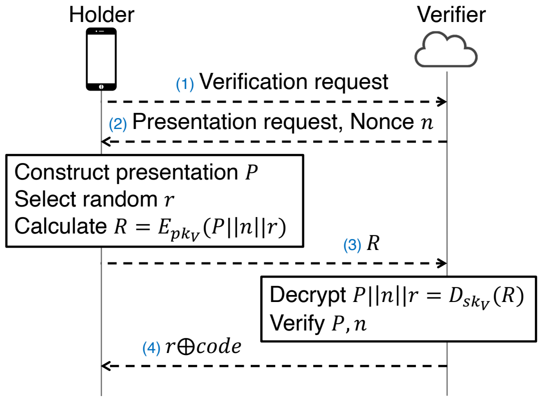

Holder
Verifier
(1) Verification request
(2) Presentation request, Nonce n
Construct presentation P
Select random r
Calculate $R = E_{pk_V}(P||n||r)$
(3) R
Decrypt $P||n||r = D_{sk_V}(R)$
Verify P, n
(4) $r \oplus code$

Figure 3: Credential presentation protocol.

The verifier operates on behalf of the callee. The verifier can be deployed in the cloud or on-premises, or even on user's telephone device. We assume that the service is public available through the Internet and the server is in the same security domain as the user device.

4.2.2 Credential Presentation

When Alice wants to make a phone call to Bob for the first time as an unknown caller, she first approaches to Bob's verifier by sending a verification request, as shown in Figure 3. Verifier will generate a *presentation request* from Bob's authentication policies and send it to Alice. Formally, the presentation request is the disjunction of M policies $PR = Pl_1 \vee Pl_2 \vee \cdots \vee Pl_M$. Alice is verified if her credential presentation satisfies any of the policies. Each policy is a set of attributes combined using predicate logics including equality, inequality, conjunction, disjunction, set membership check, and range check. Alice checks if her credentials satisfy the presentation request, creates the credential presentation using zero-knowledge proofs and sends it to Bob for authentication.

Zero-knowledge proofs. Alice can prove she has the credentials containing required attributes without revealing the credentials by using non-interactive zero-knowledge proofs. We consider Schnorr protocol [35] as an example of zero-knowledge protocols. Schnorr protocol is based on discrete logarithm assumption, with a group $\mathbb{G}$ of prime order p and generator g as commonly agreed parameters. Assuming Alice has a secret sk which she wants to convince Bob her knowledge that satisfies $pk = g^{sk}$, Schnorr protocol works as follows: 1) Alice chooses a random number $r \leftarrow \mathbb{Z}_p$, and sends $a = g^r$ as an announcement to Bob. 2) Bob chooses a random number $c \leftarrow \mathbb{Z}_p$ as a challenge and sends it to Alice. 3) Alice calculates $z = r + c \cdot sk$ and sends it to Bob. Bob checks $g^z = a \cdot pk^c$ to get convinced without learning anything about sk.

Using Fiat-Shamir heuristic [36], we can turn Schnorr protocol into non-interactive zero-knowledge protocol under random oracle model [37], where a cryptographic hash function H is used as a random oracle. Instead of the interactive messages, Alice calculates $c = H(pk, a = g^r), z = r + c \cdot sk$, and sends (a, z) to Bob as a proof of knowledge of sk. Bob calculates $c = H(pk, a)$ and checks $g^z = a \cdot pk^c$. We define $\pi = (a, z)$ and express the zero-knowledge proof of knowledge (zk-PoK) of sk as $\pi \in PoK\{(sk) : pk = g^{sk}\}$.

Besides Schnorr protocol, constructions of proofs of equality, inequality, conjunction, disjunction, set membership and range have been proposed in the literature [38–46]. We omit the details of these proofs but only demonstrate the concepts using the example of *holder binding*. In UCBlocker, m_1 is reserved for holder's secret key. To prove that multiple credentials are issued to the same holder, we can construct a proof using an equality composition as $\pi \in PoK\{(m_1, r_1, r_2) : y_1 = g_1^{m_1} h_1^{r_1} \wedge y_2 = g_2^{m_1} h_2^{r_2}\}$, where m_1 is the secret key sk, y_1, y_2 are blinded public keys, r_1, r_2 are two random numbers and (g_1, h_1), (g_2, h_2) are the public keys of the credentials. The proof contains two sub proofs, and the conjunction proves that the two public keys y_1, y_2 are generated from the same private key.

Presentation construction. Suppose Alice holds the credential $C = (A, B, e, s, \{m_i\}_{i \in [1,...,L]})$ that satisfies one of Bob's policy Pl, and $\mathcal{D}$ are the indexes of the attributes that will be revealed to Bob. The hidden attributes are $\{m_i\}_{i \notin \mathcal{D}}$. Alice will use some random numbers to blind the credential so that the presentation is not linkable to the credential. To do so, Alice chooses two random numbers $r_1 \leftarrow \mathbb{Z}_p^*$ and $r_2 \leftarrow \mathbb{Z}_p$, and sets $A' = A^{r_1}, \bar{A} = A'^{-e} B^{r_1}$, $r_3 = \frac{1}{r_1}$ and $B' = B^{r_1} h_0^{-r_2}, s' = s - r_2 r_3$. Then she generates a proof

$$\begin{aligned} &\pi \in PoK\{(\{m_i\}_{i \notin \mathcal{D}}, e, r_2, r_3, s') : \\ &\quad \bar{A}/B' = A'^{-e} h_0^{r_2} \wedge\ g_1 \prod_{i \in \mathcal{D}} h_i^{m_i} = B'^{r_3} h_0^{-s'} \prod_{i \notin \mathcal{D}} h_i^{-m_i}\} \end{aligned} \tag{3}$$

The credential presentation Alice sends to Bob will be $P = (A', \bar{A}, B', \pi)$. The proof contains two subproofs, where the first proves the validity of π and the second proves the validity of the hidden attributes. More detailed discussion of the construction of π is available in [31].

To verify the credential presentation, Bob's verifier checks: 1) $A' \neq 1_{G_1}$ where 1_{G_1} is the identity of G_1, 2) $e(A', w) = e(\bar{A}, g_2)$, which is zk-PoK of A and 3) π, which is zk-PoK of $(\{m_i\}_{i \notin \mathbb{D}}, e, r_2, r_3, s')$.

Please be noted that until now the presentation protocol only shows the predicate that Alice is in possession of an anonymous credential, and she can selectively disclose some of the attributes in the credential. For more advanced attribute predicates, the proof construction protocols mentioned above can be utilized. In case the requested attributes are split across multiple credentials, the holder will need to prove that these credentials are issued to the same secret key using the equality composition of zero-knowledge proof protocols.

4.2.3 Verification Code Distribution

Upon successful verification of the credential presentation, the verifier generates one or more verification codes and sends them to both the caller and callee. These codes function as a one-time password, serving to confirm the authentication during the call setup. The number of codes to be issued for each policy can be defined by the callee, with multiple codes issued at once to reduce future authentication efforts. The cached codes allow caller to make calls without another prior authentication, reducing additional call setup latency. The format of the codes can vary depending on their intended use, and may be a sequence of random numbers, or follow the E.164 standard [47] if using the caller ID as the code carrier to avoid potential blocking by telephone networks. Verifier sends the code to caller through data channel (message 2d in Figure 1 and message 4 in Figure 3). Verifier also sends the code to callee through any secure channels (message 2e in Figure 1), and callee is not required to have Internet access during caller authentication phase.

4.3 Security Analysis

We assume the cryptography building blocks are secure under corresponding security assumptions.

Credential security. During the credential issuance stage, the anonymous credential C should be transferred securely. Both the issuer and the holder must be authenticated with each other. A secure channel is assumed to be set up for this, such as Alice accessing the School website through a secure TLS connection using her username and password. If a secure channel is not present, the issuer and holder can use an authenticated encryption method to transfer the credential, making it impervious to theft by an outside party such as Eve.

Presentation security. The presentation is safeguarded against *replay attacks* using a nonce n. The verifier sends n to the holder along with the presentation request as depicted in Figure 3. The presentation, nonce, and a random number are encrypted using verifier's public key. Verifier then decrypts and confirms n to ensure that the presentation has not been replayed.

Verification code security. The verification codes should be sent securely. UCBlocker uses public-key authenticated encryption schemes, e.g. Elliptic Curve Integrated Encryption Scheme (ECIES) [48] to protect the verification codes. Holder chooses a random number r as a one-time pad, which is encrypted using verifier's public key pk_V with ECIES scheme together with the credential response P and the nonce n from verifier: $R = E_{pk_V}(P||n||r)$. Holder sends the presentation re-

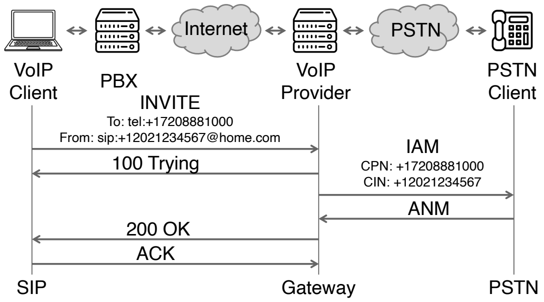


Figure 4: VoIP to PSTN call flow.

sponse R to verifier. Verifier decrypts R to get P, n, r and sends the encrypted verification code $r \oplus code$ to holder. Because of the encryption, both the presentation and the verification code are secure against *eavesdropping attacks*.

5 Call Session Binding

After call authentication, the next challenge is how to bind the authentication result (i.e., the verification code the caller received) to telephone call sessions. This is crucial as authentication occurs on the Internet, while call session signaling takes place in telephone networks. As previously discussed in Section 3.5, there are several options to bind the authentication to a call session. We explore several methods and discuss their feasibility in this paper: carrying the verification code in an existing field of signaling protocol messages, adding a new field in signaling messages, and transmitting the code through the voice channel instead of the control channel.

5.1 Caller ID Carrying the Code

Caller ID can be a viable option as it can traverse different telephone networks. As we have discussed, caller ID doesn't guarantee the caller's identification, which is leading to an increase in spam and scam calls. In UCBlocker, we discover that this weakness of caller ID can be turned into an authentication opportunity—we can re-purpose it to carry the verification code. This code serves as a binary signal that verifies the caller's authenticity and ensures the call is legitimate, replacing the phone number that is potentially spoofed.

We first analyze the feasibility of the *Caller ID Carrying the Code* method in different networks.

VoIP networks. Customizing caller ID in VoIP networks is straightforward. The call flow between a VoIP client and a PSTN client is shown in Figure 4 [49]. VoIP networks use SIP signaling protocol, while PSTN networks use SS7, where Integrated Services Digital Network (ISDN) User Part (ISUP) is used on level 4 for controlling telephone calls and for maintenance of the network. The VoIP service provider has a gateway that translates SIP messages to SS7 messages and vice versa. The caller ID can be set by the VoIP client or the connected PBX in the `From` header field of the SIP INVITE message. In the example, the `From` header contains a SIP URI with a telephone number in it. At the SIP/ISUP gateway, the INVITE message is translated into an Initial Address Message (IAM), which contains the Called Party Number (CPN) and the Calling Party Number (CIN). Though CIN is not a mandatory ISUP parameter, it is present in most of the gateway implementations.

Cellular and PSTN networks. Customizing Caller ID is more challenging in cellular and PSTN networks compared to VoIP networks. In cellular networks, the SIM card's pre-installed keys are used for subscriber authentication, and the phone number is stored by Mobile Network Operators (MNOs) in the core networks, who set the Caller ID for signaling messages. In 4G VoLTE networks, the signaling protocol SIP is used in the IP Multimedia System (IMS), but Kim et al. [50] reported flaws in the VoLTE implementation that allowed clients to set the `From` header field in the INVITE message, potentially leading to spoofed Caller IDs. However, exploiting the flaws does not align with our goal because the implementation vulnerabilities can be fixed and exploiting them may have legal restrictions. In PSTN networks, the Caller ID is inserted by the carrier's local exchange, making it difficult to customize.

Feasibility evaluation. We assess the possibility of sending verification codes through caller ID in VoIP networks. We establish a Private Branch Exchange (PBX) system and obtain a SIP trunk service with *Calling Line Identification Presentation (CLIP) no screening* capability. Then, we make a call from a SIP client and set the caller ID to a random telephone number. We successfully receive the altered caller ID without issues.

Due to the mandatory implementation of STIR/SHAKEN, the majority of VoIP service providers in the US no longer support the use of arbitrary caller IDs. However, it should be noted that customizing the caller ID itself is not against the law in the US. We believe that the *Caller ID Carrying the Code* method remains a viable option because of its simplicity. This method is appealing to international operators who are not obligated or motivated to invest in STIR/SHAKEN. Additionally, for VoIP service providers and MVNOs looking to promote their anti-spam capabilities, this lightweight method could be attractive.

5.2 Adding a Header Field

The second option is to add a header field in signaling messages for the verification code. This is exactly what STIR/SHAKEN implements where an `Identity` header is added to the SIP INVITE message. Unlike the simple verification code, the `Identity` header is complex, which contains the attestation level of the caller ID and the signature from the originating carrier. However, implementing STIR/SHAKEN requires *substantial* investment from *all* phone service and gateway service providers. Although STIR/SHAKEN can be mandatory in the US, as an end-to-end solution, it is unrealistic for UCBlocker to expect all stakeholders to cooperate in changing the signaling protocols for it to function.

5.3 Voice Channel Carrying the Code

In addition to the signaling channel, another option is to use the voice channel to transmit the verification code, which has been demonstrated by Reaves et al. in AuthLoop [12]. Compared to their effort to transfer a standard authentication protocol like TLS, using the voice channel to carry a secret code is expected to be more time efficient.

Feasibility evaluation. We use the Audio Modem Communication Library [51] to verify the feasibility of transmitting the verification code through voice channel. The software version is v1.15.4. We connect two laptops with an audio cable, both running Ubuntu 18.04. We limit the data transmission rate to 1 kbps to resemble the 500 bps data channel implemented in Authloop. The channel parameters are: carrier frequency 2000 Hz, sampling frequency 8 kHz, symbol modulation 2-QAM. The channel successfully transferred a 60kB random data file in 490s. For a 128 bits verification code, if we use the 500 bps channel and add the header and footer the same as Authloop, the estimated time cost will be about 300 ms.

5.4 Security Analysis

The verification code is a one-time password (OTP) generated randomly. In the *Caller ID Carrying the Code* method, the caller ID may be transmitted unprotected. Eve may actively monitor the signaling messages, eavesdrop the caller ID and place another call to race the legitimate caller. However, as discussed by Rubin in [52], active attacks needs resources beyond the abilities of most attackers. In our case, the active attacker must have a faster route to reach the callee so that she can win the race. Further, UCBlocker can reject concurrent or consecutive calls from the same caller ID and notice the caller so that the race attack is discovered.

The entropy of the OTP depends on the number of bits it consists of and the quality of the pseudo-random number generator (PRNG) used to generate it. When using caller ID to transmit the OTP, the entropy is subject to phone number regulations, such as the North American Numbering Plan (NANP) [53], which has rules for reserved phone numbers. The NANP phone number format can be summarized in the ten-digit notation `NXX NXX-XXXX`, where `N` denotes any of the digits `2-9`, and `X` denotes any digit `0-9`. These rules reduce the entropy of the generated random phone numbers from approximately $log_2 10^{10} \approx 2^{33}$ to about 2^{32}.

6 Evaluation

In this section, we present our prototype implementation and the evaluation results.

6.1 Prototype Implementation

We create a prototype including two sub-systems: the anonymous-credential-based authentication system comprises an issuer, a holder, a verifier, and a public ledger; UCBlocker client implemented as an application which sends and checks the verification code, and initiates and blocks calls.

The implementation of the anonymous credentials is based on Relic toolkit [54] and libpabc [55]. BBS+ blind signatures using BLS12-381 Elliptic Curve [56] is implemented. To encrypt the verification code at the end of the presentation phase, we use an public-key authenticated encryption based on libsodium [57]. We use Hyperledger Indy [58] with a local pool as the public ledger.

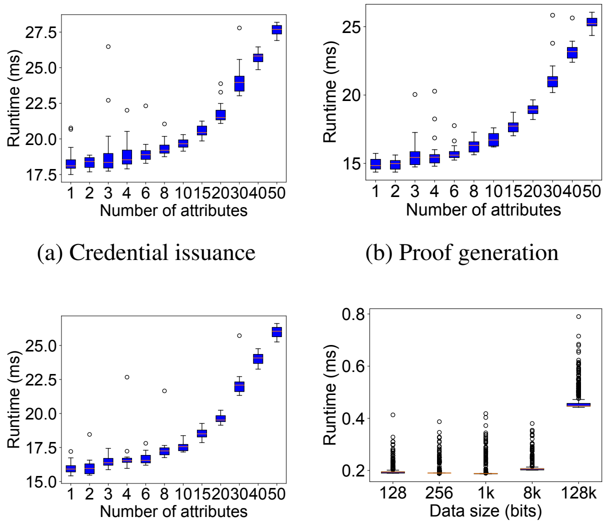
We set up a VoIP PBX using Asterisk 18 [59]. The PBX is running on an Amazon Web Services (AWS) instance with a t3.small configuration. where the operating system is Ubuntu 18.04 LTS. UCBlocker clients connect to the PBX through a Virtual Private Network (VPN), and the PBX connects to the telephone networks using SIP trunk services. We tested GoTrunk as well as Nextiva for the feasibility. They both deliver VoIP calls to our Verizon phone without issues. However, neither of them support the Clip no Screening feature anymore because of the deployment of STIR/SHAKEN.

UCBlocker client is implemented in Python, which contains a digital wallet that stores the anonymous credentials, the program instances of the issuer, the holder and the verifier. The verifier is deployed on the same computer as the VoIP client, instead of using a cloud service. It is feasible as long as the verifier is in the same security domain as the callee and is reachable through the Internet. UCBlocker also integrated an command line VoIP client to set the caller ID. The UCBlocker client is running on a MacBook with 2.6 GHz 6-Core Intel Core i7 processor, 16 GB memory, and macOS version 13.1.

6.2 Evaluation

The evaluation results are far from comprehensive because our purpose of this evaluation is to demonstrate the feasibility of the proposed solution. Through this evaluation, we are trying to make the point that the caller authentication process does not add significant delays to call setup.

Credential issuance. Our first experiment measures the credential issuance time. We measure the time from the instant when the issuer received the credential request to the moment of the credential is successfully generated. Credential issuance is a one-time operation, and it is done before the authentication. It will not introduce latency to the authentication process or the call setup process. We measure the



(a) Credential issuance

(b) Proof generation

(c) Verification

(d) Public key encryption

Figure 5: Time cost for anonymous credential issuance, proof construction, proof verification, and public key encryption.

issuance time for various numbers of attributes from 1 to 50. It is less possible for a person to have a credential with too many attributes. The results are shown in Figure 5a. The median values are from about 18 milliseconds to 27 milliseconds. It takes more time for more attributes, but the difference is not significant.

Proof generation. Figure 5b demonstrates the results of time consumption to generate a proof from a single credential. The median values are around from 15 to 25 milliseconds for attribute number distribution from 1 to 50. Proof construction with predicates from multiple credentials needs more exponentiation and multiplication and are expected to cost more time. The proof construction time will be added to the total call setup latency for unknown calls. The results prove that proof construction does not introduce significant delays.

Proof verification. The verifier checks the proof before issue a verification code to the caller. Figure 5c shows the results. The median time for verification is from 16 to 26 milliseconds, which shows that proof verification is not a main contributor to call setup latency.

Public key encryption. The caller needs to send a one-time password to the verifier so that the verification code can be issued securely. This is done using public-key authenticated encryption. The time cost for encryption of different sizes of random bits using libsodium is demonstrated in Figure 5d. The public-key authenticated encryption is fast with running time less than 1ms.

End-to-end delay. We measure the end-to-end delay of the authentication process. The total latency is about 1.5 seconds including DID lookup (400ms), proof construction (20ms), proof verification (20ms), verification code issuance (20ms), public key encryption and decryption (1ms), and communication time (1000ms in local network). Communication delays may vary for different network conditions.

Please be noted that the time cost of transmitting and checking the verification code in telephone networks is negligible. If we use the caller ID to carry the code, the process is exactly the same as using a real phone number to make a call, so there will be no delays at the caller side. At the callee side, the only additional effort is to check if the verification code is in a local database. The time cost is trivial to match a record from hundreds of records for ordinary users.

7 Discussion

We discuss the design choice of UCBlocker by considering several common questions.

Who will be issuers? During the initial bootstrapping stage, UCBlocker can establish an MVNO to act as the issuer, thereby mitigating trust and scalability concerns that arise when a large number of issuers need to be involved. However, this approach presents a privacy challenge as users are required to disclose all their attributes to the MVNO, which serves as the sole trusted entity. In the long run, as the credential issuance infrastructure is developed, UCBlocker will incorporate newly launched issuers/schemas, alleviating the burden on the MVNO as the exclusive issuer. This integration will address the privacy issue and distribute the responsibility among multiple trusted parties.

What are the differences between AC infrastructure used in UCBlocker and existing PKI? The DID scheme does not depend on centralized registries for identifiers or centralized certificate authorities for key management. The Blockchain-Based PKI is a new architecture called Decentralized Public Key Infrastructure (DPKI) [60]. DPKI architecture is distributed, self-sovereign and transparent. However, it is still developing and lack maturity.

Why do we need an anonymous credential instead of just a certificate? A certificate contains an identity and a public key. To present the certificate, both the identity and public key will be exposed. The holder does not have the choice to selectively reveal only the attributes needed for the verification. Further, the public key is a link point, where the holder's activity will be traced by multiple presentation of the certificate. For example, Alice holds an digital driver's license and she wants to call Bob. She may be willing to reveal her zip code to prove her location, but not her home address or date of birth. Anonymous credentials, on the other hand, offer selective disclosure and unlinkability. Callers won't be discouraged to make a call by privacy leakage risk.

What if the anonymous caller is a scammer? Is the anonymous caller accountable for misbehaviours? The anonymous credential protocols can incorporate accountability through cooperation between the issuer (I), verifier (V), and an authorized third party (E), such as law enforcement.

During issuance, I verifies the holder's identity, sends it to E, and E encrypts the identity using its public key. The encrypted identity is then added as a backdoor to the credential by I. During presentation, V must verify the backdoor. The backdoor can be decrypted by E using its private key when necessary. For simplicity, the design details are not discussed here. Further information about accountability can be found in the work by Camenisch et al. [61].

Are the credentials revocable? Yes. There are various solutions for designing revocable anonymous credentials, with accumulator-based solutions [62] being the most widely used. An accumulator is a single value that accumulates all revoked credentials. The accumulator value is independent from the revoked credentials list size. The issuer updates the accumulator on the ledger when a credential is revoked. To present a credential, the holder must retrieve a non-membership proof and present it to the verifier to demonstrate that her credential is still valid. In the accumulator design of [62], no operation is linearly dependent on the number of current or total deleted members. Instead, all operations only need to read and compute data that are linear in the number of changes since last read, but not in the total number of changes.

Is it legal to customize the caller ID? It depends on the laws of the jurisdiction in question. In the US, the FCC's rules under the Truth in Caller ID Act prohibit the transmission of misleading or inaccurate caller ID information with the intent to defraud, cause harm, or wrongly obtain something of value. Illegally spoofing calls can result in penalties of up to $10,000 per violation. However, spoofing is not always illegal and there are legitimate uses for it, such as when a doctor calls a patient from their personal mobile phone and displays the office number instead of their personal number or when a business displays its toll-free call-back number.

How does it work with those who don't use UCBlocker? If the person being called is not a user of UCBlocker, the caller will not be able to get authenticated and will have to resort to traditional phone calling. If the caller is not a UCBlocker user and the person being called is, then the caller won't be able to reach them without the proper credentials. The decision to use UCBlocker to block unwanted calls is at the discretion of the person being called. The caller can choose to join UCBlocker and obtain the necessary credentials if they want to make the call.

Is UCBlocker compatible with STIR/SHAKEN? The target of STIR/SHAKEN is caller ID spoofing attacks. The scheme links every call to a digital signature of the originating carrier, which signs the caller ID, callee ID and an attestation level. The call is given one of three attestation levels (A, B, or C). The *Caller ID Carrying the Code* method may result in randomly generated caller IDs being assigned level "C". UCBlocker will ignore the attestation level. It evaluates the call by matching the caller ID (verification code) to local records. In situations where the originating carrier masks the customized caller ID or even blocks the calls (instead of giving a level "C"), then we will have to choose other binding methods like using the voice channel.

How is it compatible with legitimate caller ID spoofing use cases? Legitimate caller ID spoofing replace the really phone number with another legitimate number for various purposes. Apparently, UCBlocker users does not need to hide their real phone numbers. For the purpose of providing a call-back number, this number can be sent in the credential presentation phase and is mapped to a verification code.

The callee has to publish his policies of accepted phone calls. Does it leak the callee's privacy? When Callee sets the policies, there is a balance between policy efficacy and information disclosure. The more specific the policies, the more information about Callee's preferences is revealed. There is also a trade-off between disclosing Caller's information and Callee's preferences. For instance, if Callee sets a policy as `zip code is 20001`, an attacker might deduce Callee's home location, but nothing about Caller's location is revealed because the predicate is checked on Caller side which returns only True or False. On the other hand, Callee can choose to protect his location information by setting the policy to `disclose zip code`. However, this policy will force Caller to reveal her location, which might discourage her from making the call.

8 Related Work

Technical solutions for combating spam calls focus on addressing the problem of caller ID spoofing. These solutions aim to either authenticate the caller's identity or attest to the authenticity of the caller ID.

One approach is to use network-based solutions that attest whether the caller ID is spoofed or not. For example, the STIR/SHAKEN protocol [7, 8] is a standardized solution that has been deployed by the telecom industry in the US to address this issue. However, these types of solutions can only prevent or signal caller ID spoofing but do not protect users from unwanted calls from spammers and scammers using anonymous but legitimate caller IDs. Another approach is to use end-to-end solutions that authenticate the caller's identity before the signaling messages are sent. These solutions rely on extra channels to authenticate the caller at the callee's side, such as the voice channel [12], low-bitrate data channels [13], or normal internet connections [14, 15]. Authentication through voice channel introduces significant delays because of the low bandwidth. The data-channel-based methods fail to provide a security mechanism to bind the authentication and the call session. Moreover, these methods are caller-ID-based, which means they may protect users from caller ID spoofing, but not from other type of unwanted calls.

Caller spoofing detection has been implemented in several countries through mandatory SIM-card registration. This process necessitates mobile users to provide their real names and personal details when signing up for phone services, potentially enabling extensive public surveillance. However, it is worth noting that there are currently no laws in the US supporting mandatory SIM-card registration. Despite the existence of caller spoofing detection, it does not effectively prevent unwanted calls when attackers utilize legitimate caller IDs. In contrast, UCBlocker offers an attribute-based approach that empowers users to define what constitutes "unwanted" calls based on their own criteria. This personalized approach allows users to establish policies and preferences for call acceptance, providing a more flexible and customizable solution compared to traditional caller spoofing detection methods.

Earlier works to combat phone spam that are not caller-authentication-based are classified by Tu et al. [3] into three categories: Call Request Header Analysis, Voice Interactive Screening, and Caller Compliance. Call Request Header Analysis filters calls based on the header information associated with the call request including Caller ID Blacklisting and Whitelisting [63], Caller Reputation System [64], Caller Behavior Analysis [65], Device Fingerprinting [66] and more. Voice Interactive Screening forces the caller to interact with a voice input-based interactive system to decide if the call is spam after analyzing the caller's interaction. Published solutions include Audio Fingerprinting [67], Speech Content Analysis [68], Acoustic Pattern Analysis [69], CAPTCHA/-Turing Test [70], etc. Caller Compliance requires the caller to first satisfy a compliance requirement prior to or during a call request, including Do Not Call Registry [5], Graylisting [71], Call Back Verification [72], Weakly Secret Information [73], Payment at Risk [74], to name some of them.

9 Conclusion

We present UCBlocker, an end-to-end unwanted call blocking system that is based on anonymous credentials. One novel idea in our design is to authenticate the caller based on attributes instead of the phone number, which makes unwanted call blocking possible. Another novel ideal is the decoupling of the caller authentication process and the call initiation process using the verification code. This design achieves our goal to minimize changes to the telephone networks, minimize extra call initiation delays, and eliminate the requirement on call-time data channel. The design of UCBlocker brings forth a significant potential to foster the restoration of public trust in telephone networks and promote increased utilization of phone calls.

Acknowledgement

This work was supported in part by the US National Science Foundation under grants 1916902, 2154929, 2247560, 2247561, and 2247562, the Office of Naval Research under grant N00014-19-1-2621, and DARPA under contract HR001120C0155.

References

[1] The New York Times. Polling in Real Time: The 2018 Midterm Elections. https://www.nytimes.com/interactive/2018/upshot/elections-polls.html. Accessed 15-Jan-2023.

[2] Truecaller. Truecaller Insights 2022 U.S. Spam & Scam Report. https://www.truecaller.com/blog/insights/truecaller-insights-2022-us-spam-scam-report, 2022. Accessed 15-Jan-2023.

[3] Huahong Tu, Adam Doupé, Ziming Zhao, and Gail-Joon Ahn. Sok: Everyone hates robocalls: A survey of techniques against telephone spam. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 320–338. IEEE, 2016.

[4] Huahong Tu, Adam Doupé, Ziming Zhao, and Gail-Joon Ahn. Users really do answer telephone scams. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1327–1340, 2019.

[5] FTC. National Do Not Call Registry. https://www.donotcall.gov/, 2023. Accessed 15-Jan-2023.

[6] FTC. How To Block Unwanted Calls. https://consumer.ftc.gov/articles/how-block-unwanted-calls, 2021. Accessed 15-Jan-2023.

[7] Signature-based Handling of Asserted information using toKENs (SHAKEN). https://access.atis.org/apps/group_public/download.php/67436/ATIS-1000074.v003.pdf, 2022. Accessed 15-Jan-2023.

[8] C Wendt and M Barnes. Personal Assertion Token (PaSSporT) Extension for Signature-based Handling of Asserted information using toKENs (SHAKEN). Technical report, IETF, 2019.

[9] Jürgen Quittek, Saverio Niccolini, Sandra Tartarelli, Martin Stiemerling, Marcus Brunner, and Thilo Ewald. Detecting SPIT calls by checking human communication patterns. In *2007 IEEE International Conference on Communications*, pages 1979–1984. IEEE, 2007.

[10] Merve Sahin, Marc Relieu, and Aurélien Francillon. Using chatbots against voice spam: Analyzing {Lenny's} effectiveness. In *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*, pages 319–337, 2017.

[11] Ram Dantu and Prakash Kolan. Detecting Spam in VoIP Networks. *SRUTI*, 5:5–5, 2005.

[12] Bradley Reaves, Logan Blue, and Patrick Traynor. AuthLoop:End-to-End cryptographic authentication for telephony over voice channels. In *25th USENIX Security Symposium*, pages 963–978, 2016.

[13] Bradley Reaves, Logan Blue, Hadi Abdullah, Luis Vargas, Patrick Traynor, and Thomas Shrimpton. AuthentiCall: Efficient identity and content authentication for phone calls. In *26th USENIX Security Symposium*, pages 575–592, 2017.

[14] Jikai Li, Fernando Faria, Jinsong Chen, and Daan Liang. A mechanism to authenticate callerID. In *World Conference on Information Systems and Technologies*, pages 745–753. Springer, 2017.

[15] Ya Chen, Yazhe Wang, Yu Wang, Mingxuan Li, Guochao Dong, and Chao Liu. CallChain: Identity Authentication Based on Blockchain for Telephony Networks. In *2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 416–421. IEEE, 2021.

[16] Muhammad Ajmal Azad, Samiran Bag, Shazia Tabassum, and Feng Hao. Privy: Privacy preserving collaboration across multiple service providers to combat telecom spams. *IEEE transactions on emerging topics in computing*, 8(2):313–327, 2017.

[17] Scott E Fahlman. Selling interrupt rights: A way to control unwanted e-mail and telephone calls [Technical forum]. *IBM Systems Journal*, 41(4):759–766, 2002.

[18] Jaeseung Song, Hyoungshick Kim, and Athanasios Gkelias. iVisher: Real-Time Detection of Caller ID Spoofing. *ETRI Journal*, 36(5):865–875, 2014.

[19] Haotian Deng, Weicheng Wang, and Chunyi Peng. Ceive: Combating caller id spoofing on 4g mobile phones via callee-only inference and verification. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, pages 369–384, 2018.

[20] ITU-T. Q.700 : Introduction to CCITT Signalling System No. 7. https://www.itu.int/rec/T-REC-Q.700. Accessed 15-Jan-2023.

[21] IEEE. SIP: Session Initiation Protocol. https://www.ietf.org/rfc/rfc3261.txt. Accessed 15-Jan-2023.

[22] ITU-T. H.323 : Packet-based multimedia communications systems. https://www.itu.int/rec/T-REC-H.323. Accessed 15-Jan-2023.

[23] Christina Garman, Matthew Green, and Ian Miers. Decentralized anonymous credentials. *Cryptology ePrint Archive*, 2013.

[24] Siamak F Shahandashti and Reihaneh Safavi-Naini. Threshold attribute-based signatures and their application to anonymous credential systems. In *International conference on cryptology in Africa*, pages 198–216. Springer, 2009.

[25] Michael Backes, Jan Camenisch, and Dieter Sommer. Anonymous yet accountable access control. In *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, pages 40–46, 2005.

[26] W3C. Decentralized Identifiers (DIDs) v1.0. https://www.w3.org/TR/did-core/. Accessed 15-Jan-2023.

[27] W3C. Verifiable Credentials Data Model v1.1. https://www.w3.org/TR/vc-data-model/. Accessed 15-Jan-2023.

[28] 9to5mac. Here's where you can use your iPhone as your digital driver's license or ID in 2023. https://9to5mac.com/2023/05/22/digital-id-drivers-license-iphone-states-airports/, 2023. Accessed 15-May-2023.

[29] Kim Cameron. The laws of identity. *Microsoft Corp*, 12:8–11, 2005.

[30] W3C. Verifiable Credentials JSON Schema Specification. https://w3c-ccg.github.io/vc-json-schemas/v1/index.html. Accessed 15-Jan-2023.

[31] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong diffie hellman assumption revisited. In *International Conference on Trust and Trustworthy Computing*, pages 1–20. Springer, 2016.

[32] Hyperledger Foundation. https://www.hyperledger.org/. Accessed 15-Jan-2023.

[33] Wilfrid Hodges. Elementary predicate logic. In *Handbook of philosophical logic*, pages 1–129. Springer, 2001.

[34] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology—CRYPTO'91: Proceedings*, pages 129–140. Springer, 2001.

[35] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.

[36] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology—CRYPTO'86: Proceedings 6*, pages 186–194. Springer, 1987.

[37] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[38] Berry Schoenmakers. Lecture notes cryptographic protocols, 2022.

[39] Fabrice Boudot. Efficient proofs that a committed number lies in an interval. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 431–444. Springer, 2000.

[40] Stefan Brands. Rapid demonstration of linear relations connected by boolean operators. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 318–333. Springer, 1997.

[41] Jan Camenisch and Markus Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 107–122. Springer, 1999.

[42] Jan Camenisch, BRICS, and Markus Michels. Separability and efficiency for generic group signature schemes. In *Advances in Cryptology—CRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings*, pages 413–430. Springer, 1999.

[43] Agnes Chan, Yair Frankel, and Yiannis Tsiounis. Easy come—easy go divisible cash. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 561–575. Springer, 1998.

[44] David Chaum. Blind signatures for untraceable payments. In *Advances in cryptology*, pages 199–203. Springer, 1983.

[45] David Chaum and Torben Pryds Pedersen. Wallet databases with observers. In *Advances in Cryptology, Santa Barbara, California, USA August 16–20, 1992 Proceedings 12*, pages 89–105. Springer, 1993.

[46] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology—CRYPTO'94: 14th Annual International Cryptology Conference Santa Barbara, California, USA August 21–25, 1994 Proceedings*, pages 174–187. Springer, 2001.

[47] ITU-T. E.164 : The international public telecommunication numbering plan. https://www.itu.int/rec/T-REC-E.164/. Accessed 15-Jan-2023.

[48] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. DHAES: An Encryption Scheme Based on the Diffie-Hellman Problem. *IACR Cryptol. ePrint Arch.*, 1999:7, 1999.

[49] G Camarillo, AB Roach, J Peterson, and L Ong. Integrated services digital network (isdn) user part (isup) to session initiation protocol (sip) mapping. Technical report, IETF, 2002.

[50] Hongil Kim, Dongkwan Kim, Minhee Kwon, Hyungseok Han, Yeongjin Jang, Dongsu Han, Taesoo Kim, and Yongdae Kim. Breaking and fixing volte: Exploiting hidden data channels and mis-implementations. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 328–339, 2015.

[51] Audio Modem Communication Library. https://github.com/romanz/amodem.

[52] Aviel D Rubin. Independent one-time passwords. *computing Systems*, 9(1):15–27, 1996.

[53] NANPA. North American Numbering Plan Administrator. https://nationalnanpa.com/. Accessed 15-Jan-2023.

[54] D. F. Aranha, C. P. L. Gouvêa, T. Markmann, R. S. Wahby, and K. Liao. RELIC is an Efficient LIbrary for Cryptography. https://github.com/relic-toolkit/relic.

[55] Martin Schanzenbach and Maximilian Kaul. Privacy-preserving Attribute-based Credentials. https://github.com/Fraunhofer-AISEC/libpabc.

[56] Sean Bowe. BLS12-381: New zk-SNARK Elliptic Curve Construction. https://electriccoin.co/blog/new-snark-curve/, 2017.

[57] Libsodium. https://github.com/jedisct1/libsodium.

[58] Hyperledger Indy. https://www.hyperledger.org/use/hyperledger-indy.

[59] Asterisk. https://www.asterisk.org/.

[60] Christopher Allen. Decentralized Public Key Infrastructure. https://github.com/WebOfTrustInfo/rwot1-sf/blob/master/final-documents/dpki.pdf/, 2015. Accessed 15-May-2023.

[61] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Balancing Accountability and Privacy Using E-Cash. In *Security and Cryptography for Networks: 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006. Proceedings 5*, pages 141–155. Springer, 2006.

[62] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Crypto*, volume 2442, pages 61–76. Springer, 2002.

[63] Sharbani Pandit, Roberto Perdisci, Mustaque Ahamad, and Payas Gupta. Towards Measuring the Effectiveness of Telephony Blacklists. In *NDSS*, 2018.

[64] Vijay Balasubramaniyan, Mustaque Ahamad, and Haesun Park. CallRank: Combating SPIT Using Call Duration, Social Networks and Global Reputation. In *CEAS*. Citeseer, 2007.

[65] Payas Gupta, Bharat Srinivasan, Vijay Balasubramaniyan, and Mustaque Ahamad. Phoneypot: Data-driven understanding of telephony threats. In *NDSS*, volume 107, page 108, 2015.

[66] Hong Yan, Kunwadee Sripanidkulchai, Hui Zhang, Zon-Yin Shae, and Debanjan Saha. Incorporating active fingerprinting into spit prevention systems. In *Third annual security workshop (VSW'06)*. Citeseer, 2006.

[67] Dirk Lentzen, Gary Grutzek, Heiko Knospe, and Christoph Porschmann. Content-based detection and prevention of spam over IP telephony-system design, prototype and first results. In *2011 IEEE International Conference on Communications (ICC)*, pages 1–5. IEEE, 2011.

[68] Federico Maggi. Are the con artists back? a preliminary analysis of modern phone frauds. In *2010 10th IEEE International Conference on Computer and Information Technology*, pages 824–831. IEEE, 2010.

[69] Vijay A Balasubramaniyan, Aamir Poonawalla, Mustaque Ahamad, Michael T Hunter, and Patrick Traynor. Pindr0p: Using single-ended audio features to determine call provenance. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 109–120, 2010.

[70] Sajad Shirali-Shahreza and Ali Movaghar. A new anti-spam protocol using CAPTCHA. In *2007 IEEE International Conference on Networking, Sensing and Control*, pages 234–238. IEEE, 2007.

[71] Juergen Quittek, Saverio Niccolini, Sandra Tartarelli, and Roman Schlegel. On spam over internet telephony (SPIT) prevention. *IEEE Communications Magazine*, 46(8):80–86, 2008.

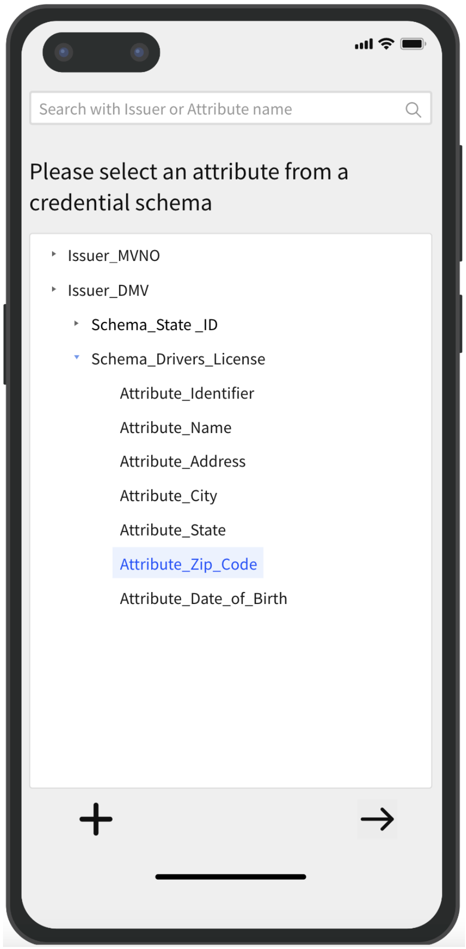
[72] Hossen Mustafa, Wenyuan Xu, Ahmad-Reza Sadeghi, and Steffen Schulz. End-to-end detection of caller ID spoofing attacks. *IEEE Transactions on Dependable and Secure Computing*, 15(3):423–436, 2016.

[73] Kumiko Ono and Henning Schulzrinne. Have I met you before? Using cross-media relations to reduce SPIT. In *Proceedings of the 3rd International Conference on Principles, Systems and Applications of IP Telecommunications*, pages 1–7, 2009.

[74] Yacine Rebahi, Dorgham Sisalem, and Thomas Magedanz. Sip spam detection. In *International Conference on Digital Telecommunications (ICDT'06)*, pages 68–68. IEEE, 2006.

A User Interfaces

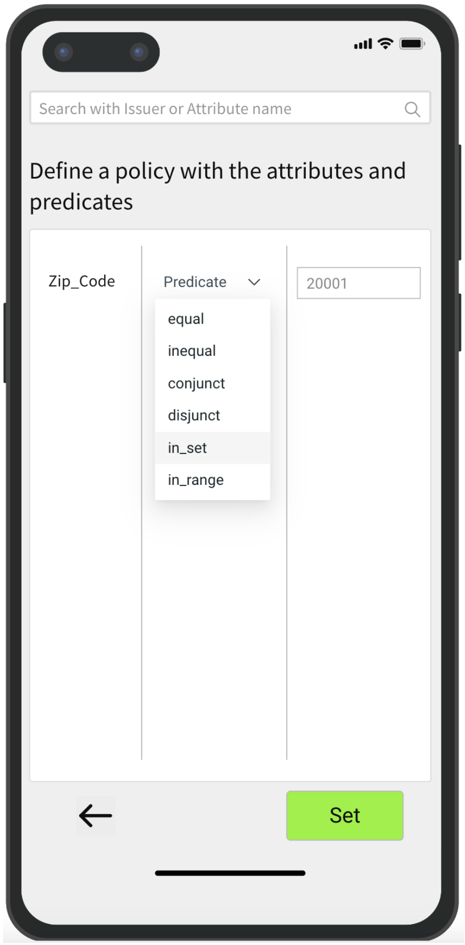
In this Appendix, we include the user interfaces to show how UCBlocker users can search and select credential schemas and attributes to create policies. We also include a UI to show how a caller can choose to expose certain attributes when making a phone call.



Please select an attribute from a credential schema

Figure 6: Policy creation: step 1.

To establish a policy, UCBlocker users can initiate a search by entering either the name of an issuer or the name of an attribute, as illustrated in Figure 6. The UCBlocker client then connects to the public ledger to conduct the search. The search results are presented in the form of a three-layered tree structure. The root layer represents the issuers, the second layer displays the schemas defined by each issuer, and the third layer comprises the attributes included within the

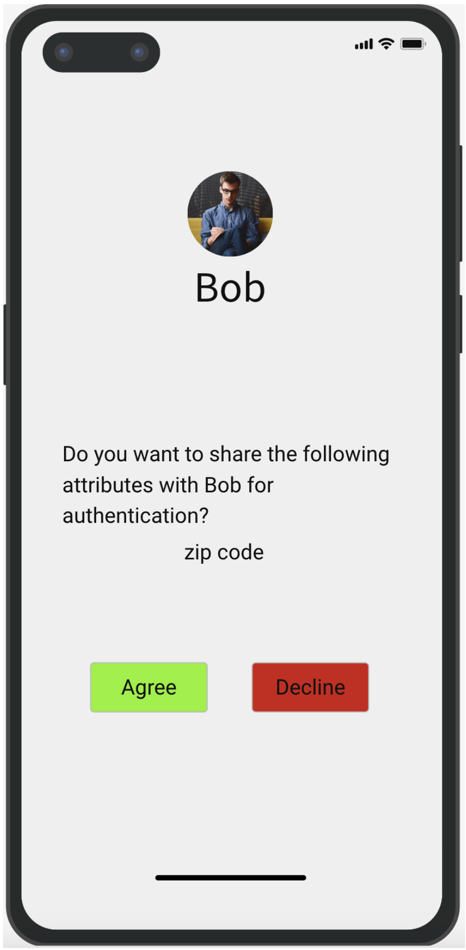


Define a policy with the attributes and predicates

Zip_Code

20001

Figure 7: Policy creation: step 2.



Bob

Do you want to share the following attributes with Bob for authentication?

zip code

Figure 8: Caller warning to reveal certain attributes.

schemas. Users can select one or more attributes from the tree and add them to the policy by clicking the "+" button. We show two issuers, namely the MVNO and a DMV, who have asserted the *zip code* attribute of the user. The MVNO, functioning as the bootstrapping issuer, can verify the users' zip code in the physical world and subsequently issue a credential. On the other hand, the DMV has published two schemas, namely the *State ID* and the *Drivers License*, both of which encompass the *zip code* attribute. In this scenario, the user opts to select the *zip code* attribute from the *Drivers License* schema, considering its more widespread usage within the United States.

In the second step, the selected attributes are combined using predicate logic to define the policy. This process involves various operations, such as equality, inequality, conjunction (AND), disjunction (OR), set membership check, and range check. These operations allow for the creation of flexible and specific policies based on the desired conditions and relationships among the chosen attributes.

As depicted in Figure 7, the user intends to establish a policy that allows phone calls solely from their actual neighbors. This policy is defined by verifying the caller's home address zip code against a selected set of zip codes representing the neighborhood. In this case, the user can choose a specific set of zip codes corresponding to the desired neighborhood and validate whether the caller's zip code falls within this set. This policy ensures that incoming calls are only accepted from individuals residing within the designated neighborhood.

During the authentication phase, UCBlocker automatically verifies the caller's credentials against the policies defined by the callee. If a match is found, UCBlocker will notify the caller that certain attributes need to be disclosed in order to successfully pass the authentication process, as shown in Figure 8. If the caller has privacy concerns regarding the disclosure of specific attributes, they have the option to decline the request and the authentication will not proceed further. This gives the caller control over their privacy and allows them to make an informed decision regarding the authentication process.