Callisto: A Cryptographic Approach to Detecting Serial Perpetrators of Sexual Misconduct

Anjana Rajan
Callisto
anjana@projectcallisto.org

Lucy Qin
Boston University
lucyq@bu.edu

David W. Archer
dwa@galois.com

Dan Boneh
Stanford University
dabo@cs.stanford.edu

Tancrede Lepoint
tancrede.lepoint@sri.com

Mayank Varia
Boston University
varia@bu.edu

ABSTRACT

Sexual misconduct is prevalent in workplace and education settings but stigma and risk of further damage deter many victims from seeking justice. Callisto, a non-profit that has created an online sexual assault reporting platform for college campuses, is expanding its work to combat sexual assault and harassment in other industries. In this new product, users will be invited to an online "matching escrow" that will detect repeat perpetrators and create pathways to support for victims. Users submit encrypted data about their perpetrator, and this data can only be decrypted by the Callisto Options Counselor (a lawyer), when another user enters the identity of the same perpetrator. If the perpetrator identities match, both users will be put in touch independently with the Options Counselor, who will connect them to each other (if appropriate) and help them determine their best path towards justice. The client relationships with the Options Counselors are structured so that any client-counselor communications would be privileged. A combination of client-side encryption, encrypted communication channels, oblivious pseudo-random functions, key federation, and Shamir Secret Sharing keep data confidential in transit, at rest, and during the matching process with the guarantee that only the lawyer ever has access to user submitted data, and even then only when a match is identified.

CCS CONCEPTS

• **Security and privacy → Social aspects of security and privacy**; **Privacy-preserving protocols**; *Privacy protections*;

ACM Reference Format:
Anjana Rajan, Lucy Qin, David W. Archer, Dan Boneh, Tancrede Lepoint, and Mayank Varia. 2018. Callisto: A Cryptographic Approach to Detecting Serial Perpetrators of Sexual Misconduct. In *COMPASS '18: ACM SIGCAS Conference on Computing and Sustainable Societies (COMPASS), June 20–22, 2018, Menlo Park and San Jose, CA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3209811.3212699

COMPASS '18, June 20–22, 2018, Menlo Park and San Jose, CA, USA

ACM ISBN 978-1-4503-5816-3/18/06…$15.00
https://doi.org/10.1145/3209811.3212699

1 INTRODUCTION

Sexual assault and harassment, especially in the workplace, is prevalent across many industries. The Equal Employment Opportunity Commission's 2016 report on workplace harassment found that almost one third of the roughly 90,000 complaints received by EEOC in FY2015 included an allegation of workplace harassment [3]. However, roughly three out of four employees interviewed who experienced harassment never talked about it with a supervisor, manager or union representative, suggesting that the EEOC's figures are perhaps quite a low estimate of the true incidence of harassment. Such under-reporting perpetuates a culture of silence that deters justice for victims. Unfortunately, under-reporting by victims and associated non-accountability of perpetrators is not the only problem to be solved. Solutions that encourage and facilitate reporting must also solve the problem of keeping reported data confidential. A victim's identity, details of incidents, and identities of perpetrators are all highly sensitive information. In the wrong hands, such information can be used to cause harm and also inhibit the empowerment for victims we seek to encourage. In addition, society can use such information to damage victim or perpetrator reputations or wellbeing. We contribute a technical solution for the problem of protecting sensitive information about victims and perpetrators in the context of a system that uses that information to enable victims to identify perpetrators in common and discover paths open to them. The resulting system gives agency to victims as they discover and pursue paths to resolution, both by facilitating information sharing and protecting that information from mis-use.

2 APPLICATION CONTEXT

Callisto is a non-profit that creates technology to combat sexual assault with a currently campus platform for college students to securely report sexual assault incidents. Reported information is held in a data escrow, and is used to notify a student's institution if another student identifies the same perpetrator. Studies showed that survivors of sexual assault who used this online platform were five times more likely to report incidents than those who did not. In addition, victims using the platform reported incidents an average of four months after the incident in contrast to an average eleven-month delay for victims not using the platform [2]. The lessons learned from this campus product demonstrate that reporting an incident of sexual assault was significantly more rapid and likely when victims knew they were not the only victim of an assailant.

Learning of another victim of the same assailant dramatically increased victims' willingness to report incidents, as well as their perceived likelihood of being believed if they reported. These learnings, together with the success of the campus product, motivate a new platform to address sexual assault and harassment underreporting in non-educational environments such as workplaces. This new platform adopts an invitation-only membership model. Partnering organizations provide Callisto with email addresses of their members. Users with those email addresses receive encrypted email invitations to activate accounts on our system and verify their identity. Once verified, users are free to submit incident reports, modify them, or delete them at will. Incident reports include the identity of the accused perpetrator, which may be in one or more of several forms: a cell phone number, a social media URL, an e-mail address, and other forms that are supported. When matches are identified, a lawyer will reach out to each victim individually (and if appropriate, may connect the group of matched victims together) to help them find their desired pathway to justice. This new platform uses a principled approach to ensure privacy of victims and perpetrators against adversaries included in our preliminary threat model.

3 TECHNICAL SOLUTION

Our technical solution approaches the problem of protecting victim (and perpetrator) privacy in this application context through comprehensive use of privacy-preserving encryption technologies, strong authentication, and best practices in system security design. Personal information of users, their accounts of incidents, and the identities of perpetrators are encrypted before they leave the user's browser and remain encrypted until they are decrypted on the personal workstation of a lawyer. That is, Callisto servers never store data about users, incidents, or victims in plaintext form. In addition, those in the lawyer role cannot access incident or perpetrator identity information *unless more than one user has identified the same perpetrator.* Access to user accounts are protected by multi-factor authentication, strong password requirements, and both e-mail and telephone contact verification. The overall system security stance is driven by the NIST Risk Management Framework [6]. In the remainder of this article, we focus on the privacy-preservation encryption technologies used to protect victim and perpetrator personal information. Submission of the personal information described above involves interaction between a reporting user's browser, an application server, a key server, and a lawyer's browser. After submission, this data is stored in and accessed via a relational database on the application server. The key server serves two roles: it stores a predetermined key whose purpose is explained below, and it authenticates users during the login process. Once available for decryption, data may be reviewed by lawyers in browsers on their personal workstations.

3.1 Cryptographic Components

The system is designed to prevent unauthorized access to information about users, incidents, and perpetrators. Since information about incidents is particularly sensitive, the following cryptographic components are used so that information about non-matched records cannot be revealed.

Shamir Secret Sharing: let s be a secret key. Shamir Secret Sharing [7] is a technique that lets us split s into many shares $s_1, s_2, \ldots, s_n$ so that a single share reveals nothing about s, but when two shares become public, anyone can reconstruct the secret s. Briefly, to create shares of s we generate a random line in a plane of possible secret shares whose y-intercept is the secret s. The shares of s are points on this line. A single point reveals nothing about the line, but two points reveal the line and thus enable computing of its y-intercept.

Oblivious pseudo-random functions (OPRFs): An OPRF uses a secret key k_s to map a value x to a pseudorandom value $\hat{x}$[1]. This secret key k_s is stored on the key server. A client who has an input x can interact with the key server to obtain $\hat{x}$, an "entropy-boosted" encoding of x. The "oblivious" property refers to the fact that in this process, the key server learns nothing about x, yet the client learns $\hat{x}$. We stress that this process is deterministic: evaluating the OPRF at the point x (using the key k_s) always results in the same pseudorandom value $\hat{x}$.

Symmetric Encryption: For a given secret key k and message m we use $c = E(k, m)$ to denote the encryption of m using key k using an authenticated encryption scheme. We use $D(k, c)$ to denote the decryption process. We use `libsodium`'s default implementation for symmetric encryption [5] in the demonstration application we describe in Section 5, but will move to a NIST-approved algorithm for our final released product.

Public key encryption: We use $c = \mathcal{E}(pk, m)$ to denote encryption of m using public key pk, and $\mathcal{D}(sk, c)$ to denote decryption of c using the corresponding secret key sk. We use `libsodium` for public key operations in our demonstration application[4], but will use a NIST-approved algorithm for our final released product.

3.2 The Data Submission and Protection Process

In this section we informally describe the use of the above components during the process of submitting incident and perpetrator identity information, matching perpetrator identities from multiple incidents, and revealing necessary information to lawyers in our organization.

Setup. The key server is initialized to hold the OPRF secret key k_s. The database server holds no secrets. To simplify our description here, we describe a single lawyer. That lawyer generates a key pair pk and sk for a public-key encryption scheme and makes the public key pk available to the public.

Submitting an Incident Record. The user's browser (the client) collects details of an incident from the user and formats that data into a serialized record structure denoted `Record`. This structure contains the user's identity U and the perpetrator's identity P, along with other details of the incident.

Next, the user authenticates to the key server, and once authenticated, the user's client interacts with the oblivious pseudo-random function (OPRF) system on the key server to transform the *low-entropy* perpetrator's identity P into a *pseudorandom* value $\hat{P}$ with sufficient entropy for use in our secret sharing approach. During this step, the key server learns the identity of the user, but learns

nothing about P from the user's client. Only the user's client learns $\hat{P}$.

The client then creates a secret share of the perpetrator's identity It uses $\hat{P}$ to derive three 256-bit pseudorandom quantities (a, k, π) using the key derivation function in `libsodium`. The first two quantities define a line equation $Y = aX + k$ whose y-intercept is k. The client evaluates this line equation at the point $X = U$ to obtain $s = aU + k$. The pair (U, s) is one share of a Shamir secret sharing scheme for the secret k. All arithmetic operations are performed modulo the prime $p = 2^{256} + 297$.

Finally, the client encrypts `Record` using a fresh random record key k' to obtain an encrypted record $\texttt{eRecord} = E(k', \texttt{Record})$. It then encrypts k' twice, once using the key k generated above from $\hat{P}$, and once using a user key k_U which is discussed further below:

$$c' \leftarrow E(k,\ k'), \qquad c_U \leftarrow E(k_U,\ k').$$

All these symmetric encryptions are done using authenticated encryption with associated data (AEAD) where π is used as the associated data. The client then performs one more encryption, encrypting the triple (U, s, c') under the lawyer's *public key* pk to obtain a doubly-encrypted ciphertext:

$$c = \mathcal{E}\big(pk,\ (U, s, c')\big).$$

The client authenticates to the database server and sends it the tuple

$$(\pi,\ c,\ c_U,\ \texttt{eRecord}). \tag{1}$$

The database server stores this record in its database and sends an acknowledgement to the user's browser.

On its own, tuple (1) reveals nothing about `Record`. Not even the lawyer can decrypt `eRecord`, because there is no way for them to construct the key k. Moreover, if a user submits two records about the same P, this second record will result in the same share (U, s) as the first record, and thus nothing new is revealed about P. Finally, nothing about the submission process reveals anything to the user's browser about other incidents or perpetrator identities.

The user key k_U makes it possible for the user to update the record after the initial submission, if needed. The key k_U is generated on the user's client at initial submission time and the user is asked to write down this key as a sequence of four letter words. When the user needs to update the submission, the user types in this key and the user's client uses it to decrypt the encrypted record `eRecord`. The system locates the relevant `eRecord` using π, which is derived from the perpetrator identity provided by the user using the OPRF.

Perpetrator Matching. The database server periodically performs an off-line search for multiple occurrences of the same perpetrator identity. If it finds at least two records with the same π component, it notifies the lawyer about the match. Note that matching is done without the database server having access to perpetrator identities or incident records in unencrypted form. Thus no adversary capable of penetrating the database server can learn anything about perpetrator identities or incidents from the data stored there, or from the off-line matching process.

Revealing information to the Lawyer. When the database server identifies a match, it contacts the lawyer who retrieves the relevant records and decrypts them using her secret key. If the records are from different incidents, the lawyer obtains

$$(U_1, s_1,\ c_1',\ \texttt{eRecord}_1) \qquad \text{and} \qquad (U_2, s_2,\ c_2',\ \texttt{eRecord}_2)$$

where $U_1 \neq U_2$. By combining the shares (U_1, s_1) and (U_2, s_2), the lawyer's browser can recover the secret key k and then decrypt c_1' and c_2' using this key. From this it can decrypt $\texttt{eRecord}_1$ and $\texttt{eRecord}_2$. This reveals $\texttt{Record}_1$ and $\texttt{Record}_2$ in the clear, including incident details, user identities, and perpetrator identities. The lawyer then takes appropriate steps to contact those users and begin the resolution process.

4 ADDITIONAL DETAILS

Handling Diverse Perpetrator Identifiers. Users may identify perpetrators using one or more diverse credentials such as social media URLs, phone numbers, or email addresses. In our system, we insist on the use of such (relatively) unambiguous identifiers at present. To allow for this diversity of credentials, π and s are not scalars as described above. Instead, they are *vectors* of values $\vec{\pi}$ and $\vec{s}$, where each component in these vectors corresponds to a particular predetermined type of identifying credential. We say that two records $(\vec{\pi}_1, \ldots)$ and $(\vec{\pi}_2, \ldots)$ are a match if the vectors $\vec{\pi}_1$ and $\vec{\pi}_2$ match on at least one component. Moreover, the lawyer workstation can fill in additional components in the $\vec{\pi}$ and $\vec{s}$ vectors for an incident once such a match is determined, because they have access to the necessary encryption keys to update the relevant data. Thus our system *propagates* perpetrator identities, using the human judgment of the lawyer, to achieve more complete identity credential vectors for perpetrators.

Privacy Roots of Trust. Every system has one or more *roots of trust*: one or more components that are assumed secure in certain ways. Briefly, our system assumes the following roots of trust for privacy preservation.

The user's browser (and computer) are one root of trust. We assume no adversary has compromised that component with the intent of observing interactions with our system. In other words, protecting against system adversaries with vantage point on the user's computer is *out of scope* for our system. Users are responsible for adequately protecting the passphrase they use to log in, as well as the devices and accounts they use for multi-factor authentication

Above, we described a system using a single key server. The OPRF key is a highly sensitive secret in our system. If this key is exposed to an adversary, then that adversary can unmask records in the database by performing an exhaustive search over potential perpetrator identities. In addition, if this key is lost and must be replaced, then matching post-loss perpetrator identities to pre-loss identities is impossible. To further protect the OPRF key, our system uses two servers, each of which keep a single cryptographic share of the key, but not the whole key. Thus key theft requires compromise of two distinct servers with different administrators, and possibly running different operating systems. This *split server* is another root of trust of our system. One of these servers is a dedicated, highly protected physical server. The other is a virtualized server hosted on a separate cloud provider. To prevent loss of the OPRF key, both servers are backed up in an encrypted backing store. To

further thwart dictionary attacks, the key servers perform access rate limiting.

The lawyer's workstation contains a password vault used to hold the lawyer's secret key that enables decryption of user profiles, as well as incident records and perpetrator identities (these latter two only after a match, as described above). This vault is in turn protected by a passphrase known only to the lawyer. Such passphrases are a *partial* root of trust for our system. We may increase security in this area by storing cryptographic shares of the lawyer's private key in a secret-shared fashion, and performing decryptions without ever bringing those key shares together "in the clear".

We note that the database server is *not* a privacy root of trust for our system, because it holds no secret keys, and because all sensitive information held there is encrypted with keys held on other components in our system.

Table 1 summarizes how we cryptographically protect each sensitive asset type held by the system against unauthorized access. Authorized access to data is limited. In brief, the access control policy is that users may enter, update, or delete their own data; lawyers may access certain user data, such as the user's contact information, personal data, and data that allows for tracking of progress toward each user's resolution goals; and lawyers also may access incident reports and perpetrator identity information, but only after more than one user reports an incident with the same perpetrator.

Information Asset	How Protected
Invitation e-mail address	Public key cryptosystem
Sponsor identity	Public key cryptosystem
Account outreach message	Commercial secure messaging
Account username	Hash-stretched SHA-2
User's personal data	Public key cryptosystem
User authentication	Multi-factor authentication
User passphrase	Hash stretching PBKDF2
Incident records	Shamir secret sharing
Perpetrator identity	Shamir secret sharing
Step metadata, notes	Public key cryptosystem
Counselor secret keys	Password vault
Passwords	Hash stretching PBKDF2
Authentication	Multi-factor, as above

Table 1: Summary of Protecting Sensitive Information

5 DEMO APPLICATION AND FUTURE WORK

We created a demonstration to encourage user engagement in understanding our client-side encryption and secret sharing techniques, which can be found at:

https://cryptography.projectcallisto.org.

This demonstration application does not reflect the full cryptographic functionality of the overall solution, nor does it instantiate all servers used in our solution. Instead, it models components of our technical design for the purposes of education. The final platform release will fully instantiate the cryptography solution described above. Future work also includes comprehensive risk assessment, threat model, and proofs to rigorously explore potential attacks and protect against the design's vulnerabilities, following the NIST Risk Management Framework[6].

6 CONCLUSION

The culture of silence around sexual harassment and assault should not be the status quo. The incident reporting experience should both be accessible and give agency to survivors while providing the relevant parties with the data needed to prevent assault and stop serial perpetrators. At the same time, privacy must be preserved to provide an empowering reporting process and prevent unauthorized use of reported information. A trauma-informed technical design with data privacy at its core is essential in achieving this mission. This note describes the cryptographic approach we take to ensure that privacy while facilitating the information sharing needed to empower survivors in their pursuit of justice.

7 ACKNOWLEDGEMENTS

We would like to acknowledge the Software & Application Innovation Lab (SAIL) within the Hariri Institute for Computing at Boston University, particularly Frederick Jansen and Andrei Lapets, for their work in the software implementation of this cryptographic approach. Lucy Qin and Mayank Varia are partially supported by NSF grants #1718135 and #1414119. Dan Boneh is supported by NSF, DARPA, ONR, the Simons Foundation, and a Google faculty fellowship. Opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of these organizations.

REFERENCES

[1] Jonathan Burns, Daniel Moore, Katrina Ray, Ryan Speers, and Brian Vohaska. 2017. EC-OPRF: Oblivious Pseudorandom Functions using Elliptic Curves. IACR Cryptology ePrint Archive. (2017).
[2] Callisto. 2017. Callisto Year 2. (2017). https://www.projectcallisto.org/Callisto_Year_2_highres.pdf [Access: March 16, 2018].
[3] Chai R. Feldblum and Victoria A. Lipnic. 2016. EEOC Select Task Force on the Study of Harassment in the Workplace. https://www.eeoc.gov/eeoc/task_force/harassment/. (2016). [Access: March 16, 2018].
[4] libsodium. 2018. Public Key Authenticated Encryption. (2018). https://download.libsodium.org/doc/public-key_cryptography/authenticated_encryption.html [Access: March 28, 2018].
[5] libsodium. 2018. Secret-key authenticated encryption. (2018).
[6] NIST. 2016. Risk Management Framework. (2016).
[7] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.