Implementation and Analysis of Apple's CSAM Detection System

Alessandro Baccarini
anbaccar@buffalo.edu
University at Buffalo
December 6, 2021

1. Introduction

2. Streaming threshold PSI with associated data

Symbol	Meaning
$\mathcal{U}$	Universe of hash values
$X \subseteq \mathcal{U}$	Set of distinct hash values the server has, s.t. $\lvert X\rvert = n$.
$\bar{Y} = ((y_i, id_i, ad_i))$	Triples the client has, s.t. $\left\lvert\bar{Y}\right\rvert = m, i \in [1, m]$.
$y \in \mathcal{U}$	Hash value
$id \in \mathcal{ID}$	Unique identifier of a triple
$ad \in \mathcal{D}$	Associated data of a triple
$id(\bar{Y})$	Set of id's of triples in $\bar{Y}$
$id(\bar{Y} \cap X)$	Set of id's of triples in $\bar{Y}$ whose y is also in X
$\bar{Y}_{id} \in \mathcal{ID}^m$	List of all id's in the triples in $\bar{Y}$
$\bar{Y}_{id,ad} \subseteq (\mathcal{ID} \times \mathcal{D})$	Set of id's and ad's in the triples in $\bar{Y}$
$\bar{Y}[T] \subseteq (\mathcal{U} \times \mathcal{ID} \times \mathcal{D})^{\leq m}$	The list of triples in $\bar{Y}$ whose id's are in $T \subseteq \mathcal{ID}$
$x = d$	Assignment of value d to variable x
$x \leftarrow A(\cdot)$	x is the output of a randomized algorithm A

Table 1: PSI notations.

3. Building Blocks

We define the following cryptographic primitives and their respective constructions below:

- (Enc, Dec) is a symmetric encryption scheme with key space $\mathcal{K}'$ and provides IND-CPA security (see [KL14] §3.4.2) and *random key robustness*, which states that if $k \neq k'$ are independent random keys, then $\text{Dec}(\text{Enc}(k, m), k')$ should fail with high probability. AES128-GCM satisfies both requirements.
- $\mathbb{G}_{\text{DH}}$ is a Diffie-Hellman group of prime order q with G as a fixed generator and Decision Diffie-Hellman (DDH) assumption holds. We use Group 14 with a 2048-bit modulus, and $G = 2$.
- $H : \mathcal{U} \to \mathbb{G}_{\text{DH}}$ is a hash function modeled as as random oracle. This is implemented using HMAC with SHA256, and converting the output digest to an integer $\pmod q$.
- $h : \mathcal{U} \to \{1, \ldots, \eta\}$ is a random hash. This is implemented using SHA256 and converting the output digest to an integer $\pmod \eta$.

- $H' : \mathbb{G}_{\text{DH}} \to \mathcal{K}'$ is secure key derivation function; the uniform distribution on $\mathbb{G}_{\text{DH}}$ mapped to an "almost" uniform distribution on $\mathcal{K}'$. This is implemented using HKDF with SHA256 to produce a 128-bit key.
- Shamir secret sharing on an element of $\mathcal{K}'$ to obtain shares in $\mathbb{F}_{\text{Sh}}$ for some field $\mathbb{F}_{\text{Sh}}$ that is sufficiently large such that when choosing $t+1$ random elements from $\mathbb{F}_{\text{Sh}}$, the probability of a collision is low.
- A pseudorandom function (PRF) $F : \mathcal{K}'' \times \mathcal{ID} \to \mathbb{F}_{\text{Sh}}$. This is also constructed using HMAC with SHA256, and converting the output digest to an integer $\pmod{\text{Sh}}$.

Diffie-Hellman Random Self Reducability: Let $\mathbb{G}$ be a group of prime order q with a fixed generator $G \in \mathbb{G}$, and suppose $(L, U, V) \in \mathbb{G}^3$. Then triple (L, U, V) is a **Diffe-Hellman (DH) tuple** if there exists an $\alpha \in \mathbb{F}_q$ such that $L = G^\alpha$ and $V = U^\alpha$. We work through the arithmetic of a partial random self reduction for DH tuples as follows. Given a triple $(L, T, P) \in \mathbb{G}^3$, we

- choose a random $\beta, \gamma \in \mathbb{F}_q$,
- compute $Q = T^\beta \cdot G^\gamma$ and $S = P^\beta \cdot L^\gamma$.
- output (L, Q, S)

The transformation $(L, T, P) \to (L, Q, S)$ has the following properties:

- If (L, T, P) is a DH tuple where $L = G^\alpha$, then Q is a fresh uniformly sampled element in $\mathbb{G}$, and
$$S = P^\beta \cdot L^\gamma = (T^\alpha)^\beta \cdot (G^\alpha)^\gamma = \left(T^\beta \cdot G^\gamma\right)^\alpha = Q^\alpha.$$
- If (L, T, P) is not a DH tuple, then (Q, S) is a fresh uniformly sampled pair in $\mathbb{G}^2$.

4. Threshold PSI-AD using the DH random self reduction

We now walk through every step up the warm-up tPSI-AD protocol outlined in [BBMT21]. The specific version we are implementing occurs in four phases: S-Init, C-Init, C-Gen-Voucher, and S-Process, where S and C refer to the Server and Client, respectively.

Protocol 1: S-Init

1 Remove any duplicates from X, and let $n = |X|$.
2 Construct a hash table T:
   - Let $n' \geq n$ be the size of the table, where we chose n' to be sufficiently larger than n as to minimize collisions.
   - Choose hash function $h : \mathcal{U} \to \{1, \ldots, n'\}$.
   - Insert elements of X into T, where each cell should have at most one element.
3 Choose a random nonzero $\alpha \in \mathbb{F}_q$, compute $L = G^\alpha \in \mathbb{G}_{\text{DH}}$
4 For $i = 1$ to n' do:
   - If $T[i]$ is non-empty, set $P_i = H(T[i])^\alpha \in \mathbb{G}_{\text{DH}}$, where $T[i] \in X \subseteq \mathcal{U}$, and $H : \mathcal{U} \to \mathbb{G}_{\text{DH}}$.
   - If $T[i]$ is empty, choose a random $P_i \in \mathbb{G}_{\text{DH}}$.
5 Set $pdata = (L, P_1, \ldots, P_{n'})$.

C-Init:

1. Obtain *pdata* from the server.
2. Generate $adkey \leftarrow \mathcal{K}'$ for encryption scheme $(\mathrm{Enc}, \mathrm{Dec})$.
3. Generate $fkey \leftarrow \mathcal{K}''$ for the PRF $F : \mathcal{K}'' \times \mathcal{ID} \rightarrow \mathbb{F}_{\mathrm{Sh}}$.
4. Initialize threshold Shamir secret sharing for *adkey*:

$$f(x) = a_0 + a_1 x + a_2 x + \cdots + a_{t-1} x^{t-1},$$

where $a_0 = adkey$ is the secret.

Protocol 2: C-Init

1 Obtain *pdata* from the server.
2 Generate $adkey \leftarrow \mathcal{K}'$ for encryption scheme $(\mathrm{Enc}, \mathrm{Dec})$.
3 Generate $fkey \leftarrow \mathcal{K}''$ for the PRF $F : \mathcal{K}'' \times \mathcal{ID} \rightarrow \mathbb{F}_{\mathrm{Sh}}$.
4 Initialize threshold Shamir secret sharing for *adkey*:

$$f(x) = a_0 + a_1 x + a_2 x + \cdots + a_{t-1} x^{t-1},$$

where $a_0 = adkey$ is the secret.

Protocol 3: C-Gen-Voucher

1 Encrypt *ad* as $adct \leftarrow \mathrm{Enc}(adkey, ad)$, and all *adct* must be the same length.
2 Compute $x = F(fkey, id) \in \mathbb{F}_{\mathrm{Sh}}$.
3 Generate a share $sh = (x, f(x)) \in \mathbb{F}_{\mathrm{Sh}}$ of *adkey* (guarantees duplicate triples with the same *id* will produce the same *sh*).
4 Choose a random key $rkey \leftarrow \mathcal{K}'$ and compute $rct \leftarrow \mathrm{Enc}(rkey, (adct, sh))$.
5 Compute $w = h(y) \in \{1, \ldots, n'\}$.
6 Sample random $\beta, \gamma \in \mathbb{F}_q$, and use P_w, L from *pdata* to compute:

$$Q = H(y)^{\beta} \cdot G^{\gamma} \text{ and } S = P_w^{\beta} \cdot L^{\gamma},$$

where if $y = T[w]$, then $P_w = H(y)^{\alpha}$ and $S = Q^{\alpha}$.
7 The client is applying the DH random self reduction to the triple $(L, H(y), P_w)$. If $y = T[w]$, then $P_w = H(y)^{\alpha}$ and (Q, S) satisfies $S = Q^{\alpha}$. Otherwise, (Q, S) are random elements of $\mathbb{G}_{\mathrm{DH}}$.
8 Compute $ct \leftarrow \mathrm{Enc}(H'(S), rkey)$, where $H' : \mathbb{G}_{\mathrm{DH}} \rightarrow \mathcal{K}'$.
9 Send $voucher = (id, Q, ct, rct)$ to the server.

C-Gen-Voucher:

1. Encrypt *ad* as $adct \leftarrow \mathrm{Enc}(adkey, ad)$, and all *adct* must be the same length.
2. Compute $x = F(fkey, id) \in \mathbb{F}_{\mathrm{Sh}}$.
3. Generate a share $sh = (x, f(x)) \in \mathbb{F}_{\mathrm{Sh}}$ of *adkey* (guarantees duplicate triples with the same *id* will produce the same *sh*).
4. Choose a random key $rkey \leftarrow \mathcal{K}'$ and compute $rct \leftarrow \mathrm{Enc}(rkey, (adct, sh))$.
5. Compute $w = h(y) \in \{1, \ldots, n'\}$.

6. Sample random $\beta, \gamma \in \mathbb{F}_q$, and use P_w, L from *pdata* to compute:

$$Q = H(y)^\beta \cdot G^\gamma \text{ and } S = P_w^\beta \cdot L^\gamma,$$

where if $y = T[w]$, then $P_w = H(y)^\alpha$ and $S = Q^\alpha$. The client is applying the DH random self reduction to the triple $(L, H(y), P_w)$. If $y = T[w]$, then $P_w = H(y)^\alpha$ and (Q, S) satisfies $S = Q^\alpha$. Otherwise, (Q, S) are random elements of $\mathbb{G}_{\text{DH}}$.
7. Compute $ct \leftarrow \text{Enc}\left(H'(S), rkey\right)$, where $H' : \mathbb{G}_{\text{DH}} \to \mathcal{K}'$.
8. Send $voucher = (id, Q, ct, rct)$ to the server.

S-Process:

1. Initialize empty set `SHARES` and an empty list `IDLIST`.
2. For each voucher (id, Q, ct, rct) received, do:
   - Append id to `IDLIST`.
   - Compute $\hat{S} = Q^\alpha \in \mathbb{G}_{\text{DH}}$,
   - Set $rkey = \text{Dec}(H'(\hat{S}), ct)$.
   - Set $(adct, sh) = \text{Dec}(rkey, rct)$.
   - If either decryptions "fails", y is a non-match, and ignore the voucher.
   - Otherwise, we found a match and add $(id, adct, sh)$ to `SHARES`.
3. Let t' denote the number of *unique* shares in `SHARES`, and t' should equal the size of $id(\bar{Y} \cap X)$.
   - If $t' < t$, let `OUTSET` be the set of identifiers in `SHARES`.
   - If $t' \geq t$, do:
     - Use t shares to reconstruct $adkey \in \mathcal{K}'$.
     - Initialize `OUTSET` $= \{\emptyset\}$.
     - For each triple $(id, adct, sh) \in$ `SHARES`, compute $ad = \text{Dec}(adkey, adct)$. If it fails, discard the voucher. Otherwise, add (id, ad) to `OUTSET`.
   - Output `IDLIST` and `OUTSET`.

References

[BBMT21] Abhishek Bhowmick, Dan Boneh, Steve Myers, and Kunal Talwar Karl Tarbe. The Apple PSI System. https://www.apple.com/child-safety/pdf/Apple_PSI_System_Security_Protocol_and_Analysis.pdf, 2021.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. Chapman and Hall/CRC, 2014.

A. Mathematical Reference

A.1. Finite Fields

Definition A.1. *A* (finite) finite $\mathbb{F}$ *is a set defined with operations* $+, \times$ *such that the following hold:*

- $\mathbb{F}$ *is abelian with respect to "+," where we let 0 denote the identity element.*

– $\mathbb{F} \setminus \{0\}$ *is abelian with respect to "*$\times$*," where we let 1 denote the identity element. We write* ab *in place of* $a \times b$*.*
– (Distributivity:) $\forall a, b, c \in \mathbb{F}$*, we have* $a \times (b + c) = ab + ac$

The additive inverse of $a \in \mathbb{F}$ denoted by $-a$ is a unique element that satisfies $a + (-a) = 0$, and the multiplicative inverse of $a \in \mathbb{F} \setminus \{0\}$ denoted a^{-1} is the unique element that satisfies $aa^{-1} = 1$.

The *order* of a F is the number of elements in $\mathbb{F}$, provided $\mathbb{F}$ is finite. If q is a prime power $q = p^r$ for a prime p and positive integer r, we can establish the field $\mathbb{F}_p$ of prime order q.