

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. $ X = n$.
$\bar{Y} = ((y_i, id_i, ad_i))$	Triples the client has, s.t. $ \bar{Y} = 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}_{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 $G = 2$.
- $H : \mathcal{U} \rightarrow \mathbb{G}_{\text{DH}}$ is a hash function modeled as a random oracle. This is implemented using HMAC with SHA256, and converting the output digest to an integer $(\text{mod } q)$.
- $h : \mathcal{U} \rightarrow \{1, \dots, \eta\}$ is a random hash. This is implemented using SHA256 and converting the output digest to an integer $(\text{mod } \eta)$.

- $H' : \mathbb{G}_{\text{DH}} \rightarrow \mathcal{K}'$ is secure key derivation function; the uniform distribution on \mathbb{G}_{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}_{Sh} for some field \mathbb{F}_{Sh} that is sufficiently large such that when choosing $t + 1$ random elements from \mathbb{F}_{Sh} , the probability of a collision is low.
- A pseudorandom function (PRF) $F : \mathcal{K}'' \times \mathcal{ID} \rightarrow \mathbb{F}_{\text{Sh}}$. This is also constructed using HMAC with SHA256, and converting the output digest to an integer $(\text{mod } \text{Sh})$.

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 **Diffie-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 $(L, T, P) \rightarrow (L, Q, S)$ 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

$$S = P^\beta \cdot L^\gamma = (T^\alpha)^\beta \cdot (G^\alpha)^\gamma = (T^\beta \cdot G^\gamma)^\alpha = Q^\alpha.$$

- 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(X)

- 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 n' is sufficiently larger than n as to minimize collisions.
 - Choose a hash function $h : \mathcal{U} \rightarrow \{1, \dots, n'\}$.
 - 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**
 - 5 **if** $T[i]$ is non-empty **then**
 - 6 Set $P_i = H(T[i])^\alpha \in \mathbb{G}_{\text{DH}}$, where $T[i] \in X \subseteq \mathcal{U}$, and $H : \mathcal{U} \rightarrow \mathbb{G}_{\text{DH}}$.
 - 7 **else**
 - 8 Choose a random $P_i \in \mathbb{G}_{\text{DH}}$.
 - 9 Set $pdata = (L, P_1, \dots, P_{n'})$.
-

Protocol 2: C-Init()

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

$$f(x) = a_0 + a_1x + a_2x + \dots + a_{t-1}x^{t-1},$$

where $a_0 = adkey$ is the secret.

Protocol 3: C-Gen-Voucher (y, id, ad)

1 Compute

$$adct \leftarrow \text{Enc}(adkey, ad),$$

and ensure all $adct$ must be the same length.

2 Compute $x = F(fkey, id) \in \mathbb{F}_{\text{Sh}}$.

3 Generate a share $sh = (x, f(x)) \in \mathbb{F}_{\text{Sh}}$ of $adkey$. This guarantees duplicate triples with the same id will produce the same sh .

4 Choose a random key $rkey \leftarrow \mathcal{K}'$ and compute

$$rct \leftarrow \text{Enc}(rkey, (adct, sh)).$$

5 Compute $w = h(y) \in \{1, \dots, n'\}$.

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}_{DH} .

7 Compute $ct \leftarrow \text{Enc}(H'(S), rkey)$, where $H' : \mathbb{G}_{\text{DH}} \rightarrow \mathcal{K}'$.

8 Send $voucher = (id, Q, ct, rct)$ to the server.

Protocol 4: S-Process

1 Initialize empty set SHARES and an empty list IDLIST.

2 **foreach** (id, Q, ct, rct) received **do**

3 Append id to IDLIST.

4 Compute $\hat{S} = Q^\alpha \in \mathbb{G}_{\text{DH}}$.

5 Set $rkey = \text{Dec}(H'(\hat{S}), ct)$.

6 Set $(adct, sh) = \text{Dec}(rkey, rct)$.

7 If either decryptions “fails”, y is a non-match, and ignore the voucher.

8 Otherwise, we found a match and add $(id, adct, sh)$ to SHARES.

9 Let t' denote the number of *unique* shares in SHARES, and t' should equal the size of $id(\bar{Y} \cap X)$.

10 **if** $t' < t$ **then**

11 let OUTSET be the set of identifiers in SHARES

12 **else if** $t' \geq t$ **then**

13 Use t shares to reconstruct $adkey \in \mathcal{K}'$.

14 Initialize OUTSET = $\{\emptyset\}$.

15 **foreach** triple $(id, adct, sh) \in \text{SHARES}$ **do**

16 compute $ad = \text{Dec}(adkey, adct)$.

17 If it fails, discard the voucher. Otherwise, add (id, ad) to OUTSET.

18 Output IDLIST and OUTSET.

Since the original protocol relies on slightly different primitives than the version provided in this paper, we argue its correctness below.

Theorem 4.1 (Correctness). *Suppose the client and server honestly adhere to Protocols 1, 2, 3, and 4, and the following assumptions hold:*

- (i) $H' : \mathbb{G}_{\text{DH}} \rightarrow \mathcal{K}'$ is a secure key derivation function,
- (ii) (Enc, Dec) is random key robust, and
- (iii) F is a secure PRF.

The server learns the required tPSI-AD output with high probability.

5. Implementation Details

6. Discussion

One of the major discrepancies with the system is how it handles duplicate hashes. The authors consider it a possibility for the existence of triples in \bar{Y} with the same hash y , provided they have different corresponding id 's. More specifically, it is valid for a \bar{Y} to contain the triples $t_i = (y_i, id_i, ad_i)$ and $t_j = (y_i, id_j, ad_i)$ for $i \neq j$. If a client submits a voucher derived from t_j after previously submitting t_i , we are guaranteed to treat t_j as unique match with a valid share of $adkey$, since x is derived from id , namely $x = F(fkey, id)$. This implies a client inputting multiple copies of the same violating image on their device may exceed the threshold t as if every picture was unique. This may be considered an acceptably property of the system, but is nonetheless worth noting.

7. Conclusion

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.