The Apple PSI System [Bhowmick et al., 2021]

Alessandro Baccarini

University at Buffalo

anbaccar@buffalo.edu

December 15, 2021

Table of Contents

Motivations

Why?

- August 2021 – Apple unveils plans for new Child Sexual Abuse Material (CSAM) detection system.
- Designed to automatically detect known CSAM images stored in iCloud, and report the users to authorities.
- Aimed to be packaged with iOS 15 and iPadOS 15.
- Very poorly received in media and tech communities.

Apple's CSAM detection tech is under fire — again

Zack Whittaker @zackwhittaker / 2:28 PM EDT • August 18, 2021 Comment

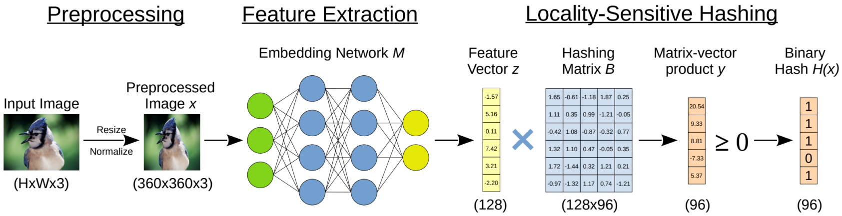
- September 2021 – Apple postpones rollout indefinitely.

What security goals do "we" want?

- Server cannot recover the user's matched photos without exceeding some threshold.
- False positives are impossible.
- No information is learned about non-matched images.
- User cannot learn any information from the CSAM database.
- User cannot identify which images were flagged as CSAM by the system.

NeuralHash

- Different from our standard notion of hash functions.
- Insensitive to small perturbations (cropping, rotation, mirroring, watermarking).



[Struppek et al., 2021]

NeuralHash

– Contains some collision-related issues [Athalye, 2021]...

 $\stackrel{?}{=}$ 

```
$ python nnhash.py cat.png
59a34eabe31910abfb06f308
$ python nnhash.py dog.png
59a34eabe31910abfb06f308
```

A Crash Course in Private Set Intersection (PSI)

- Let $\mathcal{U}$ be the universe of all possible image hashes.
- $X \subseteq \mathcal{U}$ is set of image hashes we want to match against, stored on the server.
- A client has a list of m triples

$$\bar{Y} = ((y_1, id_1, ad_1), \ldots (y_m, id_m, ad_m)) \in (\mathcal{U} \times \mathcal{ID} \times \mathcal{D})^m,$$

  where $y \in \mathcal{U}$ is the hash of an image, a unique identifier $id \in \mathcal{ID}$, and some associated data $ad \in \mathcal{D}$.
- When the protocol terminates, the server learns the identifiers and associated data of the intersection of $\bar{Y}$ and X, namely $id\left(\bar{Y} \cap X\right)$

Two PSI Protocols

Threshold PSI-AD

Add a threshold parameter t, such that if $\left|id\left(\bar{Y} \cap X\right)\right| \leq t$, the server learns only the id's. If $\left|id\left(\bar{Y} \cap X\right)\right| > t$, then the server learns the associated data for all identifiers in the intersection.

Fuzzy Threshold PSI-AD

Extension of prior scheme, but adds "synthetic matches" so the server does not know the number of matches in the intersection before the threshold t is exceeded.

Protocol Description

Server Setup

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 (minimize collisions).
   - Choose hash function $h : \mathcal{U} \to \{1, \dots, n'\}$ (SHA256 modulo n').
   - Insert elements of X into T, each cell should have at most one element.
3. Choose a random nonzero $\alpha \in \mathbb{F}_q$, compute $L = G^\alpha \in \mathbb{G}$, where $\mathbb{G}$ is a DH group modulo prime p (2048-bit) with a fixed generator $G = 2$.
4. For $i = 1$ to n' do:
   - If $T[i]$ is non-empty, set $P_i = H(T[i])^\alpha \in \mathbb{G}$, where $T[i] \in X \subseteq \mathcal{U}$, and $H : \mathcal{U} \to \mathbb{G}$ (SHA256 modulo p).
   - If $T[i]$ is empty, choose a random $P_i \in \mathbb{G}$.
5. set $pdata = (L, P_1, \dots, P_{n'})$.

Client Setup

1. Obtain *pdata* from the server.
2. Generate keys:
   - $adkey \leftarrow \mathcal{K}'$ for encryption scheme $(\mathrm{Enc}, \mathrm{Dec})$.
     - We use AES128-GCM for its "random key robustness" property.
     - $Dec(Enc(k, m), k')$ should fail, where $k \neq k'$ are independent random keys.
   - $fkey \leftarrow \mathcal{K}''$ for the PRF $F : \mathcal{K}'' \times \mathcal{ID} \rightarrow \mathbb{F}_{\mathsf{Sh}}$.
   - Initialize threshold *Shamir secret sharing* for *adkey*:

$$f(x) = a_0 + a_1 x + a_2 x + \cdots + a_t x^t,$$

   where $a_0 = adkey$ is the secret. Reconstruction involves Lagrange interpolation.

Client Voucher Generation on Input Triple (y, id, ad)

1. Encrypt ad as $adct \leftarrow \text{Enc}(adkey, ad)$, and 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 $adkey$ (guarantees duplicate triples with the same id will produce the same sh).
4. Choose a random key $rkey \leftarrow \mathcal{K}'$ and compute $rct \leftarrow \text{Enc}(rkey, (adct, sh))$.

Client Voucher Generation on Input Triple (y, id, ad)

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 $S = Q^{\alpha}$ (DH random self reduction).
7. Compute $ct \leftarrow \mathrm{Enc}\left(H'(S), rkey\right)$, where $H' : \mathbb{G} \rightarrow \mathcal{K}'$ (HKDF with SHA256).
8. Send $voucher = (id, Q, ct, rct)$ to the server.

Server Voucher Processing

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 $\hat{S} = Q^{\alpha} \in \mathbb{G}$,
   - 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 $SHARES$.

Server Voucher Processing

3. Let t' denote the number of *unique* shares in $SHARES$, and t' should equal the size of $id(\bar{Y} \cap X)$.
   - If $t' \leq t$, let $OUTSET$ be the set of identifiers in $SHARES$.
   - If $t' > t$, do:
     - Use $(t+1)$ shares to reconstruct $adkey \in \mathcal{K}'$.
     - Initialize $OUTSET = \{\emptyset\}$.
     - For each triple $(id, adct, sh) \in SHARES$, compute $ad = \mathrm{Dec}(adkey, adct)$. If it fails, discard the voucher. Otherwise, add (id, ad) to $OUTLIST$.
   - Output $IDLIST$ and $OUTSET$.

(Brief) Discussion

- Protocol is correct if the client and server adhere to the protocol (proof omitted for obvious reasons).
- Using “simpler” constructions guarantees the same level of security as the original protocol (potentially for the price of degraded performance).
- Construction naturally extends to ftPSI-AD, requires novel primitives.
  - Detectable hash functions, hashing to elliptic curves, etc.

Conclusions

Conclusions

- Presented Apple's PSI system for CSAM detection.
- The protocol is cryptographically sound, and meets the security goals specified earlier.

Conclusions

- So why is something like this still bad?
- What are the implications of this system?

References

Athalye, A. (2021).
NeuralHash Collider.
https://github.com/anishathalye/neural-hash-collider.
original-date: 2021-08-19T00:06:20Z.

Bhowmick, A., Boneh, D., Myers, S., and Tarbe, K. T. K. (2021).
The Apple PSI System.
https://www.apple.com/child-safety/pdf/Apple_PSI_System_Security_Protocol_and_Analysis.pdf.

Struppek, L., Hintersdorf, D., Neider, D., and Kersting, K. (2021).
Learning to break deep perceptual hashing: The use case neuralhash.
arXiv preprint arXiv:2111.06628.

Thank you!

Questions?