Homomorphic Authenticated Encryption Secure Against Chosen-Ciphertext Attack

Chihong Joo, *Aaram Yun*

Ulsan National Institute of Science and Technology, South Korea

Asiacrypt 2014

Motivation

Homomorphic encryption

- Allows 'homomorphic' evaluation of a function using ciphertexts.
- Active research area especially after Gentry's FHE (2009).

Application: cloud computing

- Upload FHE-encrypted data to the cloud, and erase the local copy.
- Describe a function f and its arguments to the cloud.
- Cloud homomorphically computes the ciphertext for the value of f.

Homomorphic authentication

- Upload the data to the cloud, and erase the local copy.
- When computing a function f on your data, you want to be sure that the returned function value is correct.
- Use *homomorphic signatures*, or *homomorphic MACs*.

Can we do *both*?

- Privacy and authenticity would be both important, just as in other applications
- Authenticated encryption
- How about *homomorphic* authenticated encryption?

HAE

- Homomorphic authenticated encryption (HAE)
  - Protects both privacy and authenticity
  - Encrypts/decrypts using the *secret* key
  - Allows *public* homomorphic evaluation of functions
  - A very natural primitive to consider

Gennaro-Wichs HAE?

- R. Gennaro, D. Wichs, “Fully Homomorphic Message Authenticators”, Asiacrypt 2013
- Their homomorphic MAC also satisfies privacy: thus HAE, even fully homomorphic
  - But, insecure when decryption queries are allowed

Generic composition?

- How about encrypt-then-authenticate?
- Yes, it works.
- Very recently, even *fully homomorphic* AE possible via generic composition
- Not available when this work was done

Our contributions

- A simple, somewhat homomorphic construction using EF-AGCD
- Various security definitions of HAE
  - And their relationship

Homomorphic Authenticated Encryption

Labeled program

- Label τ: a pointer to a data
- Labeled program $(f, \tau_1, \ldots, \tau_n)$
  - Description of a function f
  - Together with description of input arguments by labels

Annotating data

- Use the cloud as if a dictionary structure
- Each data m is annotated with a label τ
- Encrypt m w.r.t. τ to produce c, and send (τ, c) to the cloud
- When you want to compute a function f, describe its arguments by labels: $(f, \tau_1, \ldots, \tau_n)$; a labeled program

HAE

- $(ek, sk) \leftarrow \mathrm{Gen}(1^\lambda)$
- $c \leftarrow \mathrm{Enc}(sk, \tau, m)$
- $c \leftarrow \mathrm{Eval}(ek, f, c_1, \ldots, c_n)$
- $\mathrm{Dec}(sk, (f, \tau_1, \ldots, \tau_n), c) \rightarrow m$ or $\perp$

Our construction

DGHV

- Start from a variant of symmetric DGHV for $m \in \mathbb{Z}_Q$
  - $c = pq + rQ + m$
  - q is uniform random on $\mathbb{Z}_{q_o}$
  - r is a small noise in $(-2^{\rho}, 2^{\rho})$
  - Decryption: $m = c \bmod p \bmod Q$

DGHV

- Ciphertext $c = pq + rQ + m$ satisfies
  - c mod p contains the message + noise
  - c mod q_0 is uniform random on $\mathbb{Z}_{q_o}$
- Idea: use c mod q_0 as the secret homomorphism and put the authentication data $F_k(\tau)$ there
  - Adopting the technique from Catalano-Fiore MAC in Eurocrypt 2013

Our construction

- For $m \in \mathbb{Z}_Q$, use CRT to find c such that $c \equiv rQ+m \pmod{p}$, $c \equiv F_k(\tau) \pmod{q_0}$

Our construction

$$\begin{tikzcd} c \in \mathbb{Z}_{pq_o} \arrow[r, "\text{mod } p \text{ mod } Q"] \arrow[d, "\text{mod } q_o"'] & m \in \mathbb{Z}_Q \\ F_k(\tau) \in \mathbb{Z}_{q_o} & \end{tikzcd}$$

Our construction

$$
\begin{tikzcd}
c \in \mathbb{Z}_{pq_o} \arrow[r, "\text{mod } p \text{ mod } Q"] \arrow[d, "\text{mod } q_o"', "\text{homomorphic}"] & m \in \mathbb{Z}_Q \\
F_k(\tau) \in \mathbb{Z}_{q_o} &
\end{tikzcd}
$$

Our construction

somewhat homomorphic

$$
\begin{tikzcd}
c \in \mathbb{Z}_{pq_0} \arrow[r, "\text{mod } p \text{ mod } Q"] \arrow[d, "\text{mod } q_0"'] & m \in \mathbb{Z}_Q \\
F_k(\tau) \in \mathbb{Z}_{q_0} &
\end{tikzcd}
$$

homomorphic

Homomorphic AE

- Dec(sk, (f, τ_1, ..., τ_n), c):
  - If $f(F_k(\tau_1), \ldots, F_k(\tau_n)) \equiv c \pmod{q_0}$, then return $m \leftarrow c \bmod p \bmod Q$
  - Otherwise, return $\perp$

Indistinguishability

- IND-CPA definition: as usual
- For any Q with $\gcd(q_0, Q)=1$, our scheme is IND-CPA
  - Based on the decisional error-free approximate GCD assumption

Decisional vs Computational

- Decisional EF-AGCD is equivalent to computational EF-AGCD
  - Coron, Lepoint, Tibouchi, PKC 2014
- Therefore, our IND-CPA security is based on computational EF-AGCD

Unforgeability

- $(f, \tau_1, \ldots, \tau_n, c)$ is *not* a forgery if
  - Decryption fails, or
  - Decryption gives a value v which is constantly equal to $f(m_1, \ldots, m_n)$

Unforgeability

- $(f, \tau_1, \ldots, \tau_n, c)$ is a successful forgery if
  - $\mathrm{Dec}(sk, (f, \tau_1, \ldots, \tau_n), c) \neq \perp$, but
  - Type 1: $f(m_1, \ldots, m_n)$ is nonconstant or
  - Type 2: it is constant, but $\mathrm{Dec}(sk, (f, \tau_1, \ldots, \tau_n), c) \neq f(m_1, \ldots, m_n)$

Strong forgery

- In classical MAC, assuming Verify(sk, m, σ)=1
  - forgery: (m, σ) with new m
  - strong forgery: (m, σ) with new m, or new σ
- Strong unforgeability: it is infeasible to produce a strong forgery
- For any m, σ is 'computationally unique'

Power of Ver. Query

- Bellare, Goldreich, and Mityagin (2004)
  - If a MAC is strongly unforgeable, then it is secure with verification oracle
  - Reason: verification oracle can be easily simulated in such a case

Homomorphic strong unforgeability

- $(f, \tau_1, \ldots, \tau_n, c)$ is a strong forgery if
  - $\mathrm{Dec}(sk, (f, \tau_1, \ldots, \tau_n), c) \neq \perp$, but
  - Type 1: $\mathrm{Eval}(ek, f, c_1, \ldots, c_n)$ is nonconstant, or
  - Type 2: it is constant, but, $c \neq \mathrm{Eval}(ek, f, c_1, \ldots, c_n)$

Homomorphic strong unforgeability

- If HAE is strongly unforgeable, then Eval(ek, f, c_1, ..., c_n) is *essentially the only* valid ciphertext for $f(m_1, \ldots, m_n)$
- Just like classically, we may show that in this case the scheme is secure even when decryption oracle is available
- Modulo some technical difficulties

Relationship

$$
\begin{tikzcd}
\text{SUF-CCA} \arrow[r] \arrow[d, shift right] & \text{UF-CCA} \arrow[d] \\
\text{SUF-CPA} \arrow[r] \arrow[u, shift right] & \text{UF-CPA}
\end{tikzcd}
$$

SUF-CPA

- It is straightforward to show that our scheme is SUF-CPA using computational EF-AGCD assumption
- So, still secure even when decryption oracle is given: SUF-CCA

IND-CCA of HE

- For HE, IND-CCA is generally not possible because of malleability
- Homomorphic IND-CCA is defined but very technical

IND-CCA of HAE

- For HAE, meaningful definition of IND-CCA is natural!
  - Adversary can homomorphically modify the challenge ciphertext c^*
  - But, in order to make a decryption query, he has to declare what function was used to make the modification

IND-CCA of HAE

- If $f(m^*_0) \neq f(m^*_1)$, then homomorphically evaluate f on the challenge ciphertext c^*, to produce c' then decryption query for c' will trivially reveal the challenge bit
- In the IND-CCA definition of HAE, it is forbidden to make such a decryption query

Relationship

- Just like classically,
  IND-CPA + SUF-CPA → IND-CCA

Conclusion

- Proposed a simple construction of HAE based on EF-AGCD assumption
- Satisfies IND-CPA & SUF-CPA
- It follows that it satisfies IND-CCA & SUF-CCA

Thank you!