Public-Key Anamorphism in (CCA-secure) Public-Key Encryption and Beyond

CRYPTO24 – August, 2024

Joint work by:
G. Persiano (UNISA+Google)
Duong Hieu Phan (Telecom Paris)
Moti Yung (Google+Columbia)

Privacy as a Human Right

UDHR, Article 12: (1948)
No one shall be subjected to arbitrary interference with his privacy, family, home or ***correspondence***,...

End to End Encryption

- Cryptography has been very successful in providing tools for encrypting communication
  - The Signal protocol and app

But its success relies on an assumption that might be challenged in dictatorial states

The receiver-privacy assumption

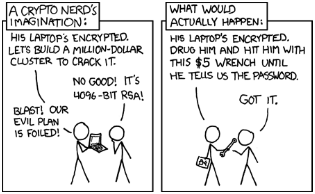
Encryption guarantees message confidentiality only with respect to parties that do not have access to the receiver's private key

The receiver-privacy assumption

The receiver keeps his secret key in a private location

Receiver privacy

- realistic for "normal" settings
- In a dictatorship, instead
  - No receiver privacy: citizens might be invited to surrender their private keys



https://xkcd.com/538/

Ban of E2E encryption

In our country, do we want to allow a means of communication between people which even in extremis, with a signed warrant from the Home Secretary personally, that we cannot read?

David Cameron
UK Prime Minister
January 2015

The Anamorphic approach for receiver privacy [PPY EC22]

- An anamorphic public key $\mathtt{aPK}$ is associated with two secret keys
  - the innocent secret key $\mathtt{ask}$
  - the double secret key $\mathtt{dkey}$
- A ciphertext carries two plaintexts
  - the innocent message $\mathtt{msg}$
  - the anamorphic message $\mathtt{amsg}$
- ...and there is **no** second key
  - $(\mathtt{aPK}, \mathtt{ask})$ indistinghishable from regular pair $(\mathtt{PK}, \mathtt{sk})$
- if the dictator asks for the secret key associated with $\mathtt{aPK}$, just surrender the innocent secret key $\mathtt{ask}$

Anamorphic Encryption Schemes: Syntax

- An anamorphic scheme AME consists of two encryption schemes:
  - the *regular* scheme $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$;
  - the *anamorphic* scheme $(\mathsf{aKG}, \mathsf{aEnc}, \mathsf{aDec})$;

  and it can be used to go around dictators that have access to the secret key

Bob deploys AME

Regular: use $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ as a regular public-key encryption scheme

Anamorphic deployment of AME for Alice

- Bob runs $(\mathtt{aPK}, \mathtt{ask}, \mathtt{dkey}) \leftarrow \mathsf{aKG}$
- $\mathtt{aPK}$ is public, $\mathtt{ask}$ is given to $\mathcal{D}$, and *double key* $\mathtt{dkey}$ is shared with Alice.
- Normal users use Enc and $\mathtt{aPK}$ to send messages to Bob.
- Alice wants to send anamorphic message $\mathtt{amsg}$
  - Alice sets innocent message $\mathtt{msg} =$ "Glory to our Leader"
  - Alice computes $\mathtt{act} \leftarrow \mathsf{aEnc}(\mathtt{dkey}, \mathtt{amsg}, \mathtt{msg})$
  - $\mathcal{D}$ computes $\mathtt{msg} \leftarrow \mathsf{Dec}(\mathtt{act}, \mathtt{ask})$
  - Bob gets $\mathtt{amsg} \leftarrow \mathsf{aDec}(\mathtt{act}, \mathtt{dkey})$

Note: Alice and Bob share $\mathtt{dkey}$ over a private channel

Anamorphic Encryption Schemes: Refined Syntax

The anamorphic scheme $(\mathsf{aKG}, \mathsf{aEnc}, \mathsf{aDec})$

- $\mathsf{aKG}(1^\lambda)$ returns:
  - public key $\mathtt{aPK}$
  - innocent secret key $\mathtt{ask}$
  - receiver double key $\mathtt{rdkey}$
  - sender double key $\mathtt{sdkey}$
  - $\mathtt{dkey} = \mathtt{rdkey} + \mathtt{sdkey}$
- $\mathsf{aEnc}(\mathtt{aPK}, \mathtt{sdkey}, \mathtt{msg}, \mathtt{amsg})$ returns $\mathtt{act}$
- $\mathsf{aDec}(\mathtt{rdkey}, \mathtt{act})$ returns $\mathtt{amsg}$
- $\mathsf{Dec}(\mathtt{ask}, \mathtt{act})$ returns $\mathtt{msg}$

Note: Bob sends $\mathtt{sdkey}$ to Alice on a private channel

Security notion

Real Game and **Anamorphic Game** are indistinguishable to $\mathcal{D}$

$\mathsf{RealG}_{\mathsf{AME},\mathcal{D}}(\lambda)$

1. Set $(\mathtt{PK}, \mathtt{sk}) \leftarrow \mathsf{KG}(1^\lambda)$
2. Return $\mathcal{D}^{\mathsf{EO}(\mathtt{PK},\cdot,\cdot)}(\mathtt{PK}, \mathtt{sk})$, where $\mathsf{EO}(\mathtt{PK}, \mathtt{msg}, \mathtt{amsg}) = \mathsf{Enc}(\mathtt{PK}, \mathtt{msg})$.

$\mathsf{AnamorphicG}_{\mathsf{AME},\mathcal{D}}(\lambda)$

1. Set $((\mathtt{aPK}, \mathtt{ask}), (\mathtt{sdkey}, \mathtt{rdkey})) \leftarrow \mathsf{aKG}(1^\lambda)$
2. Return $\mathcal{D}^{\mathsf{AO}(\mathtt{aPK},\mathtt{sdkey},\cdot,\cdot)}(\mathtt{aPK}, \mathtt{ask})$, where $\mathsf{AO}(\mathtt{PK}, \mathtt{sdkey}, \mathtt{msg}, \mathtt{amsg}) = \mathsf{aEnc}(\mathtt{aPK}, \mathtt{sdkey}, \mathtt{msg}, \mathtt{amsg})$.

The anamorphic scheme is not asymmetric anymore

- $(\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ is asymmetric:
  no secret information must be transferred from key generator to message sender

- $(\mathsf{aKG}, \mathsf{aEnc}, \mathsf{aDec})$ is symmetric:
  key generator must transfer `sdkey` *to message sender*

The Naor-Yung Encryption Scheme

Let $\mathcal{E} = (\mathsf{KG}, \mathsf{Enc}, \mathsf{Dec})$ be any encryption scheme

Regular Mode

- Bob runs KG twice, randomly selects Σ and sets $\mathtt{PK} = (\mathtt{PK}_0, \mathtt{PK}_1, \Sigma)$ and $\mathtt{sk} = \mathtt{sk}_0$
- Alice wants to send "Glory to our Leader" to Bob
  - Computes $\mathtt{ct}_0 = \mathsf{Enc}(\mathtt{PK}_0, \text{"Glory to our Leader"})$
  - Computes $\mathtt{ct}_1 = \mathsf{Enc}(\mathtt{PK}_1, \text{"Glory to our Leader"})$
  - Computes NIZK proof Π that $\mathtt{ct}_0$ and $\mathtt{ct}_1$ carry the same plaintext
  - Set $\mathtt{ct} = (\mathtt{ct}_0, \mathtt{ct}_1, \Pi)$
- Bob wants to decrypt $\mathtt{ct}$,
  - Checks Π is a valid proof
  - If valid, decrypts $\mathtt{ct}_0$ using $\mathtt{sk}$

The Naor-Yung Encryption Scheme

Anamorphic Mode

- Bob runs KG twice, runs the **simulator** to get $(\Sigma, \mathtt{aux})$ and sets $\mathtt{PK} = (\mathtt{PK}_0, \mathtt{PK}_1, \Sigma)$ and $\mathtt{sk} = \mathtt{sk}_0$
- sets $\mathtt{rdkey} = \mathtt{sk}_1$ and $\mathtt{sdkey} = \mathtt{aux}$ is shared with Alice
- Alice wants to send $\mathtt{msg} =$ "Glory to our Leader" to the dictator and $\mathtt{amsg} =$ "Fire our Leader" to Bob
  - Computes $\mathtt{ct}_0 = \mathsf{Enc}(\mathtt{PK}_0, \mathtt{msg})$ and $\mathtt{ct}_1 = \mathsf{Enc}(\mathtt{PK}_1, \mathtt{amsg})$
  - **Simulate** NIZK proof Π that $\mathtt{ct}_0$ and $\mathtt{ct}_1$ carry the same plaintext
  - Sets $\mathtt{ct} = (\mathtt{ct}_0, \mathtt{ct}_1, \Pi)$
- To decrypt $\mathtt{ct}$, Alice uses $\mathtt{sk}_1$ to decrypt $\mathtt{ct}_1$
- If asked to surrender her secret key, Alice gives $\mathtt{sk}_0$
  - The dictator verifies Π, decrypts $\mathtt{ct}_0$ and reads $\mathtt{msg} =$ "Glory to our Leader"

The Koppula-Waters Encryption Scheme CRYPTO '19

- Key Generation: $\mathsf{kw.KG}(1^\lambda)$
  - Generate 2λ pairs $(\mathtt{PK}_{bi}, \mathtt{sk}_{bi})$, $b \in \{0,1\}, i \in \{1,\ldots,n\}$
  - Randomly select $a_1,\ldots,a_n \leftarrow \{0,1\}^\lambda$ and $B \leftarrow \{0,1\}^\lambda$
  - Set $\mathsf{kw}.\mathtt{PK} = \left(B, (a_i, \mathtt{PK}_{0i}, \mathtt{PK}_{1i})_{i=1}^\lambda\right)$ and $\mathsf{kw}.\mathtt{sk} = (\mathtt{sk}_{0i})_{i=1}^\lambda$
    $\mathsf{kw}.\mathtt{sk} = (\mathtt{sk}_{0i})_{i=1}^\lambda$
- Encryption: $\mathsf{kw.Enc}(\mathsf{kw}.\mathtt{PK}, \mathtt{msg})$
  - randomly select $K \leftarrow \{0,1\}^\lambda$ and $(\mathtt{sigK}, \mathtt{vK}) \leftarrow \mathsf{Sign.KG}(1^\lambda)$
  - set $c = \mathcal{F}(K,0) \oplus \mathtt{msg}$
  - for $i = 1,\ldots,\lambda$
    - ★ $\tilde{r}_i = \mathcal{F}(K,i)$ $\tilde{r}_i = \mathcal{F}(K,i)$ and $v_i \leftarrow \{0,1\}^{\lambda-1}$
    - ★ if $K_i = 0$
      $c_{0,i} = \mathsf{Enc}(\mathtt{PK}_{0i}, 1|v_i; \tilde{r}_i)$ $c_{0,i} = \mathsf{Enc}(\mathtt{PK}_{0i}, 1|v_i; \tilde{r}_i)$, $c_{1,i} = \mathsf{Enc}(\mathtt{PK}_{1i}, 0^\lambda)$
      $c_{1,i} = \mathsf{Enc}(\mathtt{PK}_{1i}, 0^\lambda)$, $c_{2,i} = G(v_i)$
    - ★ if $K_i = 1$
      $c_{0,i} = \mathsf{Enc}(\mathtt{PK}_{0i}, 0^\lambda)$ $c_{0,i} = \mathsf{Enc}(\mathtt{PK}_{0i}, 0^\lambda)$, $c_{1,i} = \mathsf{Enc}(\mathtt{PK}_{1i}, 1|v_i; \tilde{r}_i)$
      $c_{1,i} = \mathsf{Enc}(\mathtt{PK}_{1i}, 1|v_i; \tilde{r}_i)$, $c_{2,i} = G(v_i) + a_i + B \cdot \mathtt{vK}$
  - sign $(c, (c_{0,i}, c_{1,i}, c_{2,i}))$ using $\mathtt{sigK}$

Obs0: there are 2λ public keys Obs1: dictator has only λ secret keys $\mathtt{sk}_{0i}$

Making KW19 Anamorphic

Anamorphic key generation

- keep all $\mathtt{sk}_{1i}$
  - they constitute $\mathtt{rdkey}$

Anamorphic Encryption

How to encrypt:

- $\mathtt{msg}$ = "Glory to our Leader"
- $\mathtt{amsg}$ = "Fire our Leader"

1. Use kw.Enc to encrypt $\mathtt{msg}$
2. Let i be such that $K_i = 0$
   - set $c_{1,i} = \mathsf{Enc}(\mathtt{PK}_{1i}, \mathtt{amsg})$

Note1: $\Theta(\lambda)$ messages can be sent with v.h.p.

Note2: No shared information!!!

Adding Anamorphism while preserving Asymmetry

Increasing bandwidth: Anamorphic KEM+DEM

- Encryption: $\mathsf{kw.Enc}(\mathsf{kw.}\mathtt{PK}, \mathtt{msg}, \mathtt{amsg}_1, \ldots, \mathtt{amsg}_\lambda)$
  - randomly select $\mathtt{dkey} \leftarrow \mathsf{prKG}(1^\lambda)$
  - randomly select $K \leftarrow \{0,1\}^\lambda$ and $(\mathtt{sigK}, \mathtt{vK}) \leftarrow \mathsf{Sign.KG}(1^\lambda)$
  - set $c = \mathcal{F}(K, 0) \oplus \mathtt{msg}$
  - for $i = 1, \ldots, \lambda$
    - ⋆ $\tilde{r}_i = \mathcal{F}(K, i)$ and $v_i \leftarrow \mathsf{prEnc}(\mathtt{dkey}, \mathtt{amsg}_i)$
    - ⋆ if $K_i = 0$
      $c_{0,i} = \mathsf{Enc}(\mathtt{PK}_{0i}, 1|v_i; \tilde{r}_i)$, $c_{1,i} = \mathsf{Enc}(\mathtt{PK}_{1i}, \mathtt{dkey})$, $c_{2,i} = G(v_i)$
    - ⋆ if $K_i = 1$
      $c_{0,i} = \mathsf{Enc}(\mathtt{PK}_{0i}, 0^\lambda)$, $c_{1,i} = \mathsf{Enc}(\mathtt{PK}_{1i}, 1|v_i; \tilde{r}_i)$, $c_{2,i} = G(v_i) + a_i + B \cdot \mathtt{vK}$
  - sign $(c, (c_{0,i}, c_{1,i}, c_{2,i}))$ using $\mathtt{sigK}$

CCA security and Anamorphism

- NY and KW are both CCA-secure
- both give anamorphism
- of different nature

Why?

CCA security and Anamorphism

Proving CCA security from CPA security

Security reduction

- two roles in two games
  - adversary in the CPA game
  - challenger in the CCA game
- generates a CCA public key
  - receives public key `ppk` for CPA scheme
  - produces public key `cpk` for the CCA scheme
    - without knowing the secret key `psk` associated to the input `ppk`
  - keeps a state `state` (the random coin tosses)
  - `cpk` is **indistinguishable** from a real CCA public key
- handles decryption queries
  - `state` is *functionally* a decryption query
- handles encryption queries
  - receives a CPA ciphertext `pct` carrying $\mathtt{msg}_0$ and produces a CCA ciphertext `cct` on input $\mathtt{msg}_1$

The General Plan

Observation

- there are two keys at work here and two messages
  - $\mathtt{psk}$ that recovers $\mathtt{msg}_0$ from $\mathtt{pct}$
  - $\mathtt{state}$ that recovers $\mathtt{msg}_1$ from $\mathtt{cct}$

The general plan

- generate $(\mathtt{ppk}, \mathtt{psk})$ to function as $\mathtt{sdkey}$ and $\mathtt{rdkey}$
- derive $(\mathtt{cpk}, \mathtt{state})$ from $\mathtt{ppk}$ to serve as $\mathtt{aPK}$ and $\mathtt{ask}$
- anamorphic encryption of $(\mathtt{msg}, \mathtt{amsg})$
  - encrypt $\mathtt{amsg}$ using $\mathtt{ppk} = \mathtt{sdkey}$ and obtain $\mathtt{pct}$
  - use the encryption query procedure of the reduction on input $\mathtt{msg}$ and $\mathtt{pct}$ to produce anamorphic ciphertext $\mathtt{act} = \mathtt{cct}$ carrying the two messages
- anamorphic decryption of $\mathtt{act}$
  - *extract* $\mathtt{pct}$ from $\mathtt{cct}$
  - decrypt with $\mathtt{psk} = \mathtt{rdkey}$

Not there yet...

Obstacles

- dictator wants to see the secret key
  - `state` might not look like a secret key
- we need to extract `pct` from `cct`

- these are the only two obstacles
- known reductions do satisfy the requirements

Reductions yielding Public Anamorphism

Ambiguous ciphertexts in the CCA proof

- can be *decrypted* as both challenge ciphertexts
- regular sender has negligible probability of constructing an *ambiguous* ciphertext
- reduction has some *trapdoor* `trap`
  - trapdoor associated with the CRS in NY
  - special signing key in KW
- In NY the hybrid game that uses CPA security needs `trap` to construct the challenge ciphertext in
  - `trap` must be in `sdkey`

Conclusions

What we learned

- a new notion of anamorphism
  - preserving asymmetric nature of encryption
- realized by a known scheme KP (CRYPTO '19)
- sufficient conditions on CCA proof to be turned into proof of anamorphism
- which ones give public anamorphism

Thank You