

Mind the Gap: Modular Machine-checked Proofs for one-round Key Exchange Protocols

Gilles Barthe¹

Juan Manuel Crespo²

Yassine Lakhnech³

Benedikt Schmidt¹



2



3



Goal

**Obtain modular machine-checked
proofs for key exchange protocols**

Goal

Obtain modular machine-checked proofs for key exchange protocols

implicitly authenticated
KE protocols:
Naxos, HMQV, Nets,...

Goal

Obtain modular machine-checked proofs for key exchange protocols

in eCK model¹
reductions (in ROM)
to Gap DH or CDH

implicitly authenticated
KE protocols:
Naxos, HMQV, Nets,...

¹extended Canetti-Krawczyk [LLM'07]

Goal

EasyCrypt:
verifies correctness of
each proof step

**Obtain modular machine-checked
proofs for key exchange protocols**

in eCK model¹
reductions (in ROM)
to Gap DH or CDH

implicitly authenticated
KE protocols:
Naxos, HMQV, Nets,...

¹extended Canetti-Krawczyk [LLM'07]

Goal

Generic proof to
maximize reuse for
individual protocol proofs

EasyCrypt:
verifies correctness of
each proof step

**Obtain modular machine-checked
proofs for key exchange protocols**

in eCK model¹
reductions (in ROM)
to Gap DH or CDH

implicitly authenticated
KE protocols:
Naxos, HMQV, Nets,...

¹extended Canetti-Krawczyk [LLM'07]

AKE Protocols

AKE Protocols

agent A:
 $(a, A) \leftarrow \text{genStatic}()$

agent B:
 $(b, B) \leftarrow \text{genStatic}()$

a, b, \dots : static secret keys

A, B, \dots : static public keys

AKE Protocols

agent \underline{A} :
 $(a, A) \leftarrow \text{genStatic}()$

agent \underline{B} :
 $(b, B) \leftarrow \text{genStatic}()$

session of \underline{A} :

$x \leftarrow \text{genEsk}()$
 $X \leftarrow \text{genEpk}(x, a)$

a, b, \dots : static secret keys
 x, y, \dots : ephemeral secret keys

A, B, \dots : static public keys
 X, Y, \dots : static ephemeral keys

AKE Protocols

agent \underline{A} :
 $(a, A) \leftarrow \text{genStatic}()$

agent \underline{B} :
 $(b, B) \leftarrow \text{genStatic}()$

session of \underline{A} :

$x \leftarrow \text{genEsk}()$
 $X \leftarrow \text{genEpk}(x, a)$



a, b, \dots : static secret keys
 x, y, \dots : ephemeral secret keys

A, B, \dots : static public keys
 X, Y, \dots : static ephemeral keys

AKE Protocols

agent \underline{A} :
 $(a, A) \leftarrow \text{genStatic}()$

agent \underline{B} :
 $(b, B) \leftarrow \text{genStatic}()$

session of \underline{A} :

$x \leftarrow \text{genEsk}()$
 $X \leftarrow \text{genEpk}(x, a)$

X

session of \underline{B} :

$y \leftarrow \text{genEsk}()$
 $Y \leftarrow \text{genEpk}(y, b)$

a, b, \dots : static secret keys
 x, y, \dots : ephemeral secret keys

A, B, \dots : static public keys
 X, Y, \dots : static ephemeral keys

AKE Protocols

agent A:

$(a, A) \leftarrow \text{genStatic}()$

agent B:

$(b, B) \leftarrow \text{genStatic}()$

session of A:

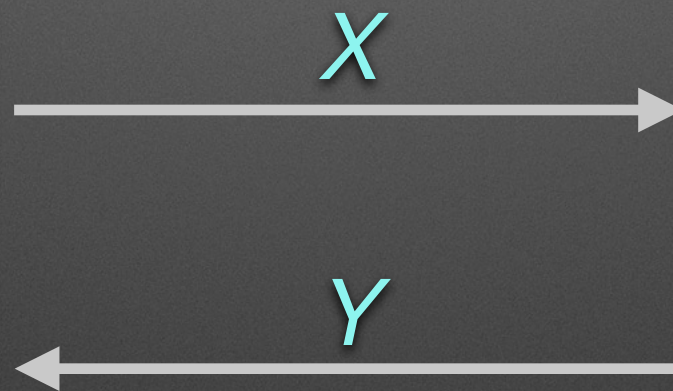
$x \leftarrow \text{genEsk}()$

$X \leftarrow \text{genEpk}(x, a)$

session of B:

$y \leftarrow \text{genEsk}()$

$Y \leftarrow \text{genEpk}(y, b)$



a, b, \dots : static secret keys

x, y, \dots : ephemeral secret keys

A, B, \dots : static public keys

X, Y, \dots : static ephemeral keys

AKE Protocols

agent A:

$(a, A) \leftarrow \text{genStatic}()$

agent B:

$(b, B) \leftarrow \text{genStatic}()$

session of A:

$x \leftarrow \text{genEsk}()$

$X \leftarrow \text{genEpk}(x, a)$

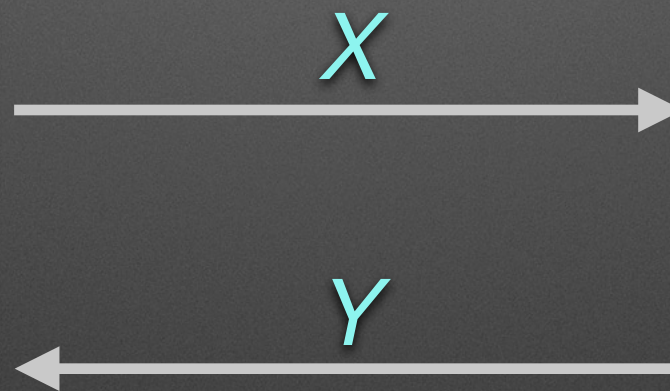
session of B:

$y \leftarrow \text{genEsk}()$

$Y \leftarrow \text{genEpk}(y, b)$

$s \leftarrow (y, X, b, A, \underline{B}, \underline{A})$

$k \leftarrow \text{sessKeyR}(s)$



a, b, \dots : static secret keys

x, y, \dots : ephemeral secret keys

A, B, \dots : static public keys

X, Y, \dots : static ephemeral keys

AKE Protocols

agent A:

$(a, A) \leftarrow \text{genStatic}()$

agent B:

$(b, B) \leftarrow \text{genStatic}()$

session of A:

$x \leftarrow \text{genEsk}()$

$X \leftarrow \text{genEpk}(x, a)$

$s' \leftarrow (x, Y, a, B, \underline{A}, \underline{B})$

$k' \leftarrow \text{sessKeyI}(s)$

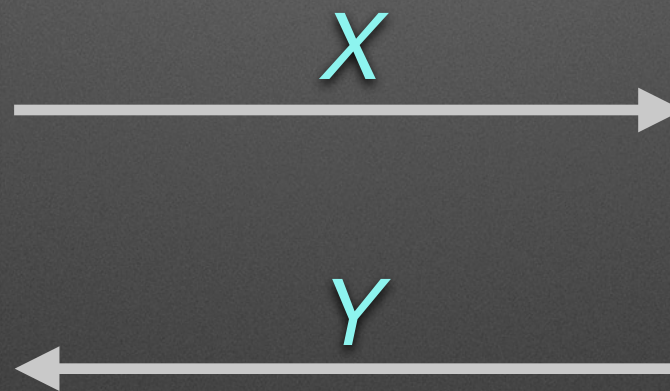
session of B:

$y \leftarrow \text{genEsk}()$

$Y \leftarrow \text{genEpk}(y, b)$

$s \leftarrow (y, X, b, A, \underline{B}, \underline{A})$

$k \leftarrow \text{sessKeyR}(s)$



a, b, \dots : static secret keys

x, y, \dots : ephemeral secret keys

A, B, \dots : static public keys

X, Y, \dots : static ephemeral keys

AKE Protocols

agent A:

$(a, A) \leftarrow \text{genStatic}()$

agent B:

$(b, B) \leftarrow \text{genStatic}()$

Correct:
both compute same key

Strong partnering:
compute same key

\Rightarrow

agree on $X, Y, \underline{A}, \underline{B}$

session of A:

$x \leftarrow \text{genEsk}()$

$X \leftarrow \text{genEpk}(x, a)$

$s' \leftarrow (x, Y, a, B, \underline{A}, \underline{B})$

$k' \leftarrow \text{sessKeyI}(s)$

session of B:

$y \leftarrow \text{genEsk}()$

$Y \leftarrow \text{genEpk}(y, b)$

$s \leftarrow (y, X, b, A, \underline{B}, \underline{A})$

$k \leftarrow \text{sessKeyR}(s)$

a, b, \dots : static secret keys

x, y, \dots : ephemeral secret keys

A, B, \dots : static public keys

X, Y, \dots : static ephemeral keys

Models: eCK^{hk} and eCK^{kr}



Queries

- `establishAgent`
- `init1`, `init2`, `resp`
- `reveal secrets keys:`
static, ephemeral, session

Models: eCK^{hk} and eCK^{kr}



Queries

- `establishAgent`
- `init1, init2, resp`
- `reveal secrets keys:`
static, ephemeral, session

eCK^{hk} : static keys honestly generated

eCK^{kr} : `establishAgent` takes public key

Models: eCK^{hk} and eCK^{kr}



Queries

- establishAgent
- $\text{init}_1, \text{init}_2, \text{resp}$
- reveal secrets keys: static, ephemeral, session

Experiment

1. perform queries
2. choose test session and obtain session or random key
3. perform queries
4. guess **real** or **random**, can only win if test is **fresh**

eCK^{hk} : static keys honestly generated

eCK^{kr} : establishAgent takes public key

Freshness in eCK

Freshness in eCK

- the test session t is fresh if the adversary does not:

Freshness in eCK

- the test session t is fresh if the adversary does not:
- reveal t 's session key

Freshness in eCK

- the test session t is fresh if the adversary does not:
- reveal t 's session key
- reveal t 's ephemeral **and** static secret key

Freshness in eCK

- the test session t is fresh if the adversary does not:
- reveal t 's session key
- reveal t 's ephemeral **and** static secret key
- register a static public key for t 's intended peer

Freshness in eCK

- the test session t is fresh if the adversary does not:
- reveal t 's session key
- reveal t 's ephemeral **and** static secret key
- register a static public key for t 's intended peer
- if there is a matching session m :

Freshness in eCK

- the test session t is fresh if the adversary does not:
- reveal t 's session key
- reveal t 's ephemeral **and** static secret key
- register a static public key for t 's intended peer
- if there is a matching session m :
 - reveal m 's session key

Freshness in eCK

- the test session t is fresh if the adversary does not:
- reveal t 's session key
- reveal t 's ephemeral **and** static secret key
- register a static public key for t 's intended peer
- if there is a matching session m :
 - reveal m 's session key
 - reveal m 's ephemeral **and** static secret key

Freshness in eCK

- the test session t is fresh if the adversary does not:
- reveal t 's session key
- reveal t 's ephemeral **and** static secret key
- register a static public key for t 's intended peer
- if there is a matching session m :
 - reveal m 's session key
 - reveal m 's ephemeral **and** static secret key
- if there is no matching session:

Freshness in eCK

- the test session t is fresh if the adversary does not:
- reveal t 's session key
- reveal t 's ephemeral **and** static secret key
- register a static public key for t 's intended peer
- if there is a matching session m :
 - reveal m 's session key
 - reveal m 's ephemeral **and** static secret key
- if there is no matching session:
 - reveal the static secret key of t 's intended peer

eCK-secure Protocols

UM+KEA:

Gap DH

$$a \in_R \mathbf{Z}_p, A = g^a$$

$$x \in_R \mathbf{Z}_p, X = g^x$$

$$\text{sessKeyl}(x, Y, a, B, \underline{A}, \underline{B}) = H(Y^x, B^a, Y^a, B^x, \underline{A}, \underline{B})$$

eCK-secure Protocols

UM+KEA:

Gap DH

$$a \in_R \mathbf{Z}_p, A = g^a$$

$$X \in_R \mathbf{Z}_p, X = g^x$$

$$\text{sessKeyl}(x, Y, a, B, \underline{A}, \underline{B}) = H(Y^x, B^a, Y^a, B^x, \underline{A}, \underline{B})$$

Naxos [LLM'07]:

Gap DH

$$a \in_R \mathbf{Z}_p, A = g^a$$

$$X \in_R \mathbf{Z}_p, X = g^{G(x,a)}$$

$$\text{sessKeyl}(x, Y, a, B, \underline{A}, \underline{B}) = H(Y^{G(x,a)}, Y^a, B^{G(x,a)}, \underline{A}, \underline{B})$$

eCK-secure Protocols

UM+KEA:

Gap DH

$$a \in_R \mathbf{Z}_p, A = g^a$$

$$X \in_R \mathbf{Z}_p, X = g^x$$

$$\text{sessKeyl}(x, Y, a, B, \underline{A}, \underline{B}) = H(Y^x, B^a, Y^a, B^x, \underline{A}, \underline{B})$$

Naxos [LLM'07]:

Gap DH

$$a \in_R \mathbf{Z}_p, A = g^a$$

$$X \in_R \mathbf{Z}_p, X = g^{G(x,a)}$$

$$\text{sessKeyl}(x, Y, a, B, \underline{A}, \underline{B}) = H(Y^{G(x,a)}, Y^a, B^{G(x,a)}, \underline{A}, \underline{B})$$

Naxos+ [Lee, Park'08]:

CDH (Trapdoor test [CKS'08])

$$a \in_R \mathbf{Z}_p, A = g^a$$

$$X \in_R \mathbf{Z}_p, X = g^{G(x,a)}$$

$$\text{sessKeyl}(x, Y, a, B, \underline{A}, \underline{B}) = H(Y^{G(x,a)}, B^a, Y^a, B^{G(x,a)}, \underline{A}, \underline{B})$$

Contribution

Contribution

- Generic Proofs¹: P_{core} secure in $M \Rightarrow T(P_{core})$ secure in eCK
 - (1) For $T =$ “hash session key of P_{core} ”
 - (2) For $T =$ “hash session key of P_{core} and use Naxos trick”

M significantly simpler than eCK

(one decisional oracle, search challenge, no freshness)

¹generalize [Kudla, Paterson'05]

Contribution

- Generic Proofs¹: P_{core} secure in $M \Rightarrow T(P_{core})$ secure in eCK
 - (1) For $T =$ “hash session key of P_{core} ”
 - (2) For $T =$ “hash session key of P_{core} and use Naxos trick”

M significantly simpler than eCK

(one decisional oracle, search challenge, no freshness)

- Instantiations for:
 - Naxos and Nets (CDH without keyReg/GDH with keyReg)
 - Naxos+ (CDH with keyReg)
 - HMQRV with $H(\sigma|X|Y|A|B)$ (GDH with keyReg)

¹generalize [Kudla, Paterson'05]

Contribution

- Generic Proofs¹: P_{core} secure in $M \Rightarrow T(P_{core})$ secure in eCK
 - (1) For $T = \text{"hash session key of } P_{core}\text{"}$
 - (2) For $T = \text{"hash session key of } P_{core} \text{ and use Naxos trick"}$

M significantly simpler than eCK

(one decisional oracle, search challenge, no freshness)

- Instantiations for:
 - Naxos and Nets (CDH without keyReg/GDH with keyReg)
 - Naxos+ (CDH with keyReg)
 - HMQRV with $H(\sigma|X|Y|A|B)$ (GDH with keyReg)
- EasyCrypt formalization

¹generalize [Kudla, Paterson'05]

Generic Proof (Naxos Trick)

Generic Proof (Naxos Trick)

1. G_0 : eCK for protocol $T^{\text{naxos}}(T^{\text{hash}}(P_{\text{core}}))$ with session key $H(s)$ and with “ephemeral exponent” $G(x,a)$.

Generic Proof (Naxos Trick)

1. G_0 : eCK for protocol $T^{\text{naxos}}(T^{\text{hash}}(P_{\text{core}}))$ with session key $H(s)$ and with “ephemeral exponent” $G(x,a)$.
2. G_1 : protocol $T^{\text{hash}}(P_{\text{core}})$ with “ephemeral exponent” x , ephemeral reveal modified to require static secret a , RO G moved into simulator.

Generic Proof (Naxos Trick)

1. G_0 : eCK for protocol $T^{\text{naxos}}(T^{\text{hash}}(P_{\text{core}}))$ with session key $H(s)$ and with “ephemeral exponent” $G(x,a)$.
2. G_1 : protocol $T^{\text{hash}}(P_{\text{core}})$ with “ephemeral exponent” x , ephemeral reveal modified to require static secret a , RO G moved into simulator.
3. G_2 : protocol P_{core} , new decisional oracle $\text{eqSess}(i,s)$ that answers “is session key of i equal to s ”, used to simulate queries $\text{sessRev}(i)$ and $H(s)$.

Generic Proof (Naxos Trick)

1. G_0 : eCK for protocol $T^{\text{naxos}}(T^{\text{hash}}(P_{\text{core}}))$ with session key $H(s)$ and with “ephemeral exponent” $G(x,a)$.
2. G_1 : protocol $T^{\text{hash}}(P_{\text{core}})$ with “ephemeral exponent” x , ephemeral reveal modified to require static secret a , RO G moved into simulator.
3. G_2 : protocol P_{core} , new decisional oracle $\text{eqSess}(i,s)$ that answers “is session key of i equal to s ”, used to simulate queries $\text{sessRev}(i)$ and $H(s)$.
4. $M_1 - M_3$: perform case distinctions on reveals, guess test/peer/actor, simulate all oracles except eqSess .

Generic Proof (Naxos Trick)

query $\text{ephemeralRev}(i,a)$ without
previous static reveal of a ?

Generic Proof (Naxos Trick)

query $\text{ephemeralRev}(i,a)$ without
previous static reveal of a ?

Yes

M_1 (eqSess orcl.)
 a, x unrevealed
must compute a

Generic Proof (Naxos Trick)

query `ephemeralRev(i,a)` without
previous static reveal of `a`?

Yes

No

static secret `b` of
test's peer revealed?

`M1` (eqSess orcl.)
`a`, `x` unrevealed
must compute `a`

Generic Proof (Naxos Trick)

query `ephemeralRev(i,a)` without
previous static reveal of `a`?

Yes

No

static secret `b` of
test's peer revealed?

No

M_1 (eqSess orcl.)
`a`, `x` unrevealed
must compute `a`

M_2 (eqSess orcl.)
`x`, `b` unrevealed
must compute `k`

Generic Proof (Naxos Trick)

query $\text{ephemeralRev}(i,a)$ without
previous static reveal of a ?

Yes

No

static secret b of
 $test$'s peer revealed?

No

Yes

M_1 (eqSess orcl.)
 a, x unrevealed
must compute a

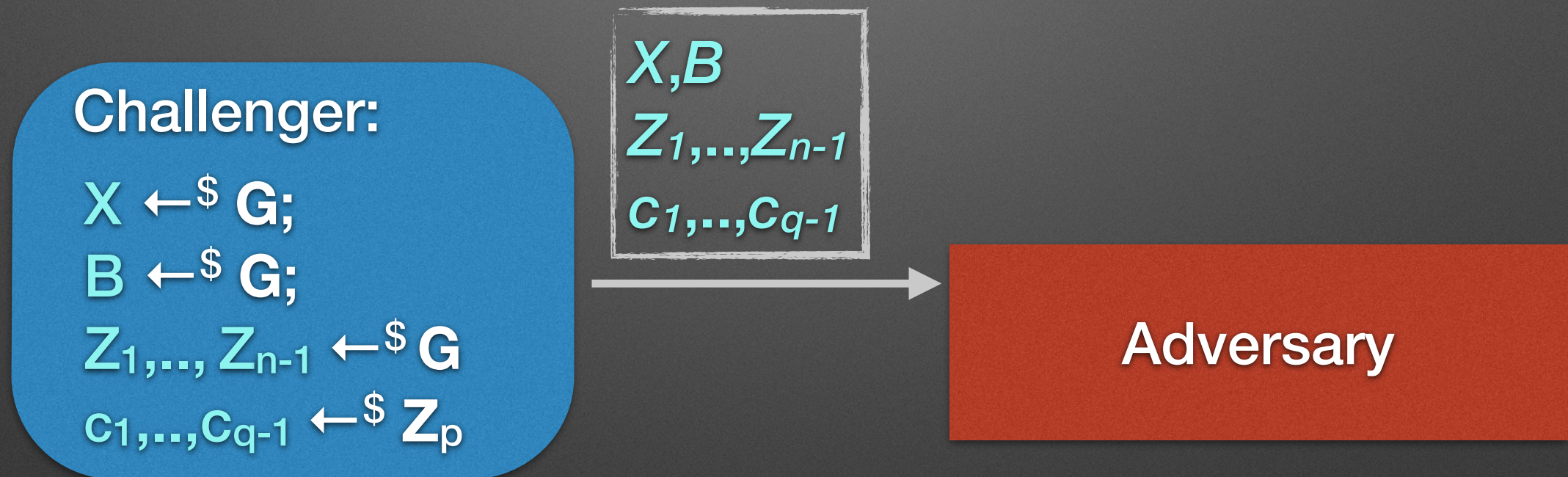
M_2 (eqSess orcl.)
 x, b unrevealed
must compute k

M_3 (no orcl.)
 x, y unrevealed
must compute k

Naxos game (no kr): M_2

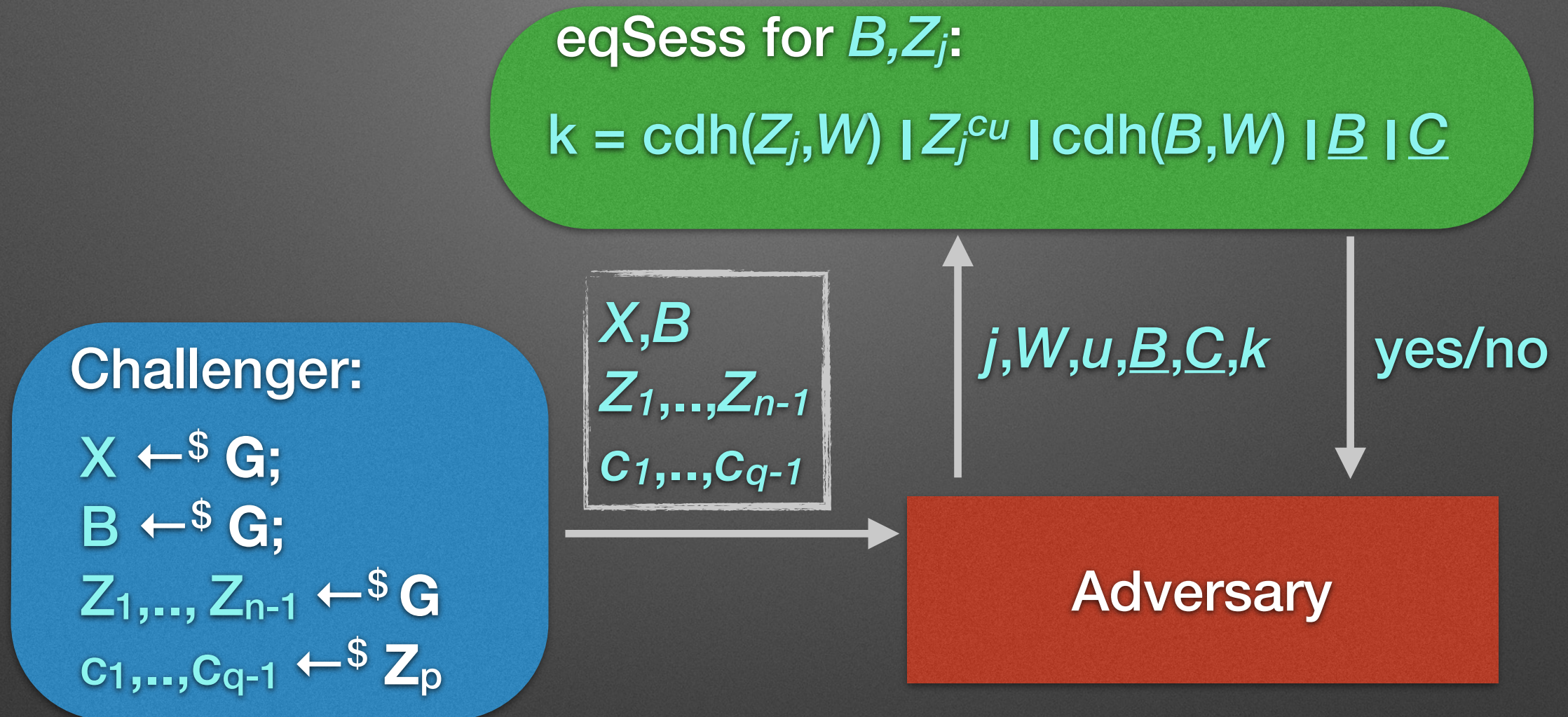
n: number of sessions
q: number of agents

Naxos game (no kr): M_2



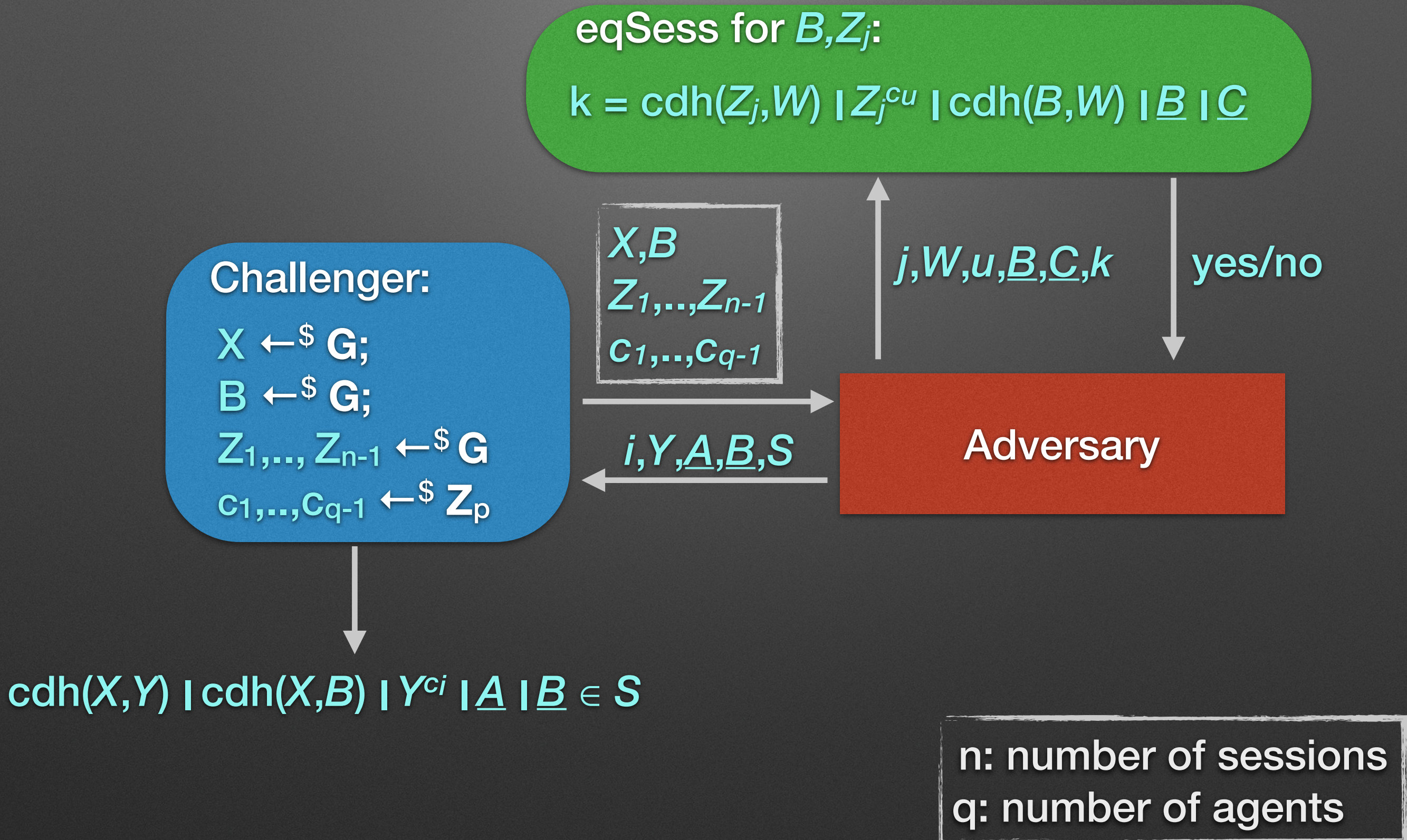
n : number of sessions
 q : number of agents

Naxos game (no kr): M_2



n : number of sessions
 q : number of agents

Naxos game (no kr): M_2



Naxos game (**with** kr): M_2

with adversarial key registration, arbitrary C instead of g^{cu}

eqSess for B, Z_j :

$$k = \text{cdh}(Z_j, W) \mid \text{cdh}(Z_j, C) \mid \text{cdh}(B, W) \mid \underline{B} \mid \underline{C}$$

Challenger:

$X \xleftarrow{\$} G;$
 $B \xleftarrow{\$} G;$
 $Z_1, \dots, Z_{n-1} \xleftarrow{\$} G$
 $C_1, \dots, C_{q-1} \xleftarrow{\$} Z_p$

X, B
 Z_1, \dots, Z_{n-1}
 C_1, \dots, C_{q-1}

$i, Y, \underline{A}, \underline{B}, S$

$j, W, C, \underline{B}, \underline{C}, k$

yes/no

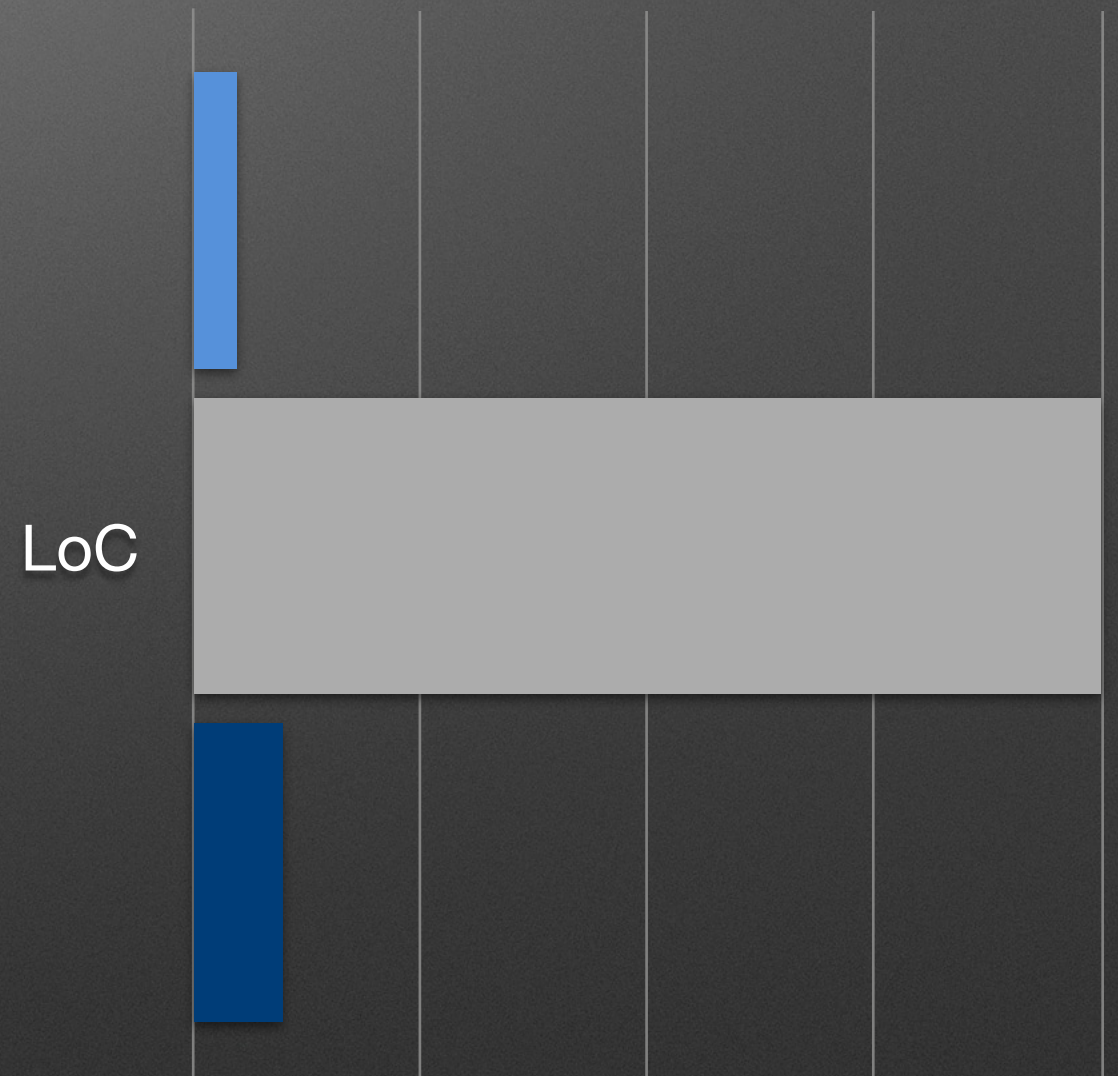
Adversary

$$\text{cdh}(X, Y) \mid \text{cdh}(X, B) \mid Y^{ci} \mid \underline{A} \mid \underline{B} \in S$$

n : number of sessions
 q : number of agents

Formalization

- EasyCrypt:
prototype version (Crypto'11)
redesigned version 1.0 (Oct. '14)
- EasyCrypt 1.0 supports generic proofs and instantiation
- Formalization:
 1. Prove generic result once and for all
 2. Prove Twin DH reductions
 2. Define concrete protocol
 3. Instantiate generic proof
 4. Protocol proof in simplified model, use Twin DH results for CDH proofs
- Future work:
develop more automation for both generic proof and instantiations



Summary

- Generic Proofs for transformations “hash session key” and “hash session key and use Naxos trick” (no kr/kr)
- Instantiations for Naxos, Nets, Naxos+, and (modified) HMQV
- EasyCrypt formalization
- Details: <http://www.easycrypt.info/ake/>

(also IACR School on Computer-Aided Crypto,
UMD, College Park, Maryland - June 1-4)

Questions?

Gap DH and Twin DH

- Gap DH: solve CDH with access to DDH oracle
- Trapdoor test:
 - Get random group element A
 - Can compute group element A' and trapdoor:
 - A' random independent of A (ignoring trapdoor)
 - for arbitrary U, V, V' can efficiently test if $\text{cdh}(A, U) = V \wedge \text{cdh}(A', U) = V'$ (using trapdoor)
- A' called a twin of A , can also generate multiple twins