On the Joint Security of Encryption and Signature in EMV

Jean Paul Degabriele, Anja Lehmann, Kenneth G. Paterson, Nigel P. Smart and Mario Strefler

CT-RSA 2012

29th February 2012

Outline

The EMV Standard

- EMV stands for Europay, Mastercard and VISA, and it is the **de facto global standard** for IC credit/debit cards – *Chip & PIN*.

- As of Q3 2011, there were more than 1.34 billion EMV cards in use worldwide.
- The standard specifies the inter-operation of IC cards with Point-Of-Sale terminals (POS) and Automated Teller Machines (ATM) .

The EMV Standard

- EMV stands for Europay, Mastercard and VISA, and it is the **de facto global standard** for IC credit/debit cards – *Chip & PIN*.

- As of Q3 2011, there were more than 1.34 billion EMV cards in use worldwide.
- The standard specifies the inter-operation of IC cards with Point-Of-Sale terminals (POS) and Automated Teller Machines (ATM) .

The EMV Standard

- EMV stands for Europay, Mastercard and VISA, and it is the **de facto global standard** for IC credit/debit cards – *Chip & PIN*.

- As of Q3 2011, there were more than 1.34 billion EMV cards in use worldwide.
- The standard specifies the inter-operation of IC cards with Point-Of-Sale terminals (POS) and Automated Teller Machines (ATM) .

EMV Cards

- EMV cards contain a ‘Chip’ which allows them to perform cryptographic computations.
- All EMV cards contain a **symmetric key** which they share with the Issuing Bank.
- Most cards are also equipped with **RSA keys** to compute signatures for card authentication and transaction authorization, and encrypt the PIN between the terminal and the card.
- The terminal authenticates the card’s public keys through its copy of the brand’s public key, and a **chain of certificates** which the card supplies.

Transaction Flow

An EMV transaction progresses over three stages:

Card Authentication: Static Data Authentication (SDA), Dynamic Data Authentication (DDA/CDA).

Cardholder Verification: paper Signature, PIN – online/offline – encrypted/cleartext.

Transaction Authorization: A successful transaction ends with the card producing a **Transaction Certificate (TC)** – a MAC computed over the transaction details.

CDA cards additionally compute a **digital signature** over the transaction details and the TC.

Transaction Flow

An EMV transaction progresses over three stages:

Card Authentication: Static Data Authentication (SDA), Dynamic Data Authentication (DDA/CDA).

Cardholder Verification: paper Signature, PIN – online/offline – encrypted/cleartext.

Transaction Authorization: A successful transaction ends with the card producing a **Transaction Certificate (TC)** – a MAC computed over the transaction details.

CDA cards additionally compute a **digital signature** over the transaction details and the TC.

Transaction Flow

An EMV transaction progresses over three stages:

Card Authentication: Static Data Authentication (SDA), Dynamic Data Authentication (DDA/CDA).

Cardholder Verification: paper Signature, PIN – online/offline – encrypted/cleartext.

Transaction Authorization: A successful transaction ends with the card producing a **Transaction Certificate (TC)** – a MAC computed over the transaction details.

CDA cards additionally compute a **digital signature** over the transaction details and the TC.

Transaction Flow

An EMV transaction progresses over three stages:

Card Authentication: Static Data Authentication (SDA), Dynamic Data Authentication (DDA/CDA).

Cardholder Verification: paper Signature, PIN – online/offline – encrypted/cleartext.

Transaction Authorization: A successful transaction ends with the card producing a **Transaction Certificate (TC)** – a MAC computed over the transaction details.

CDA cards additionally compute a **digital signature** over the transaction details and the TC.

Transaction Flow

An EMV transaction progresses over three stages:

Card Authentication: Static Data Authentication (SDA), Dynamic Data Authentication (DDA/CDA).

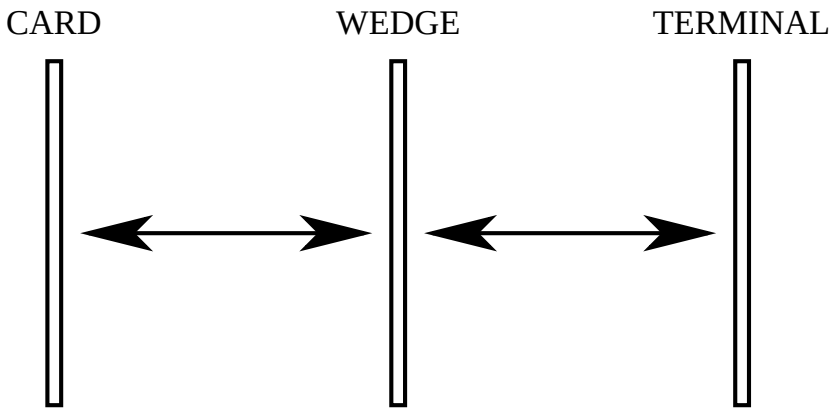
Cardholder Verification: paper Signature, PIN – online/offline – encrypted/cleartext.

Transaction Authorization: A successful transaction ends with the card producing a **Transaction Certificate (TC)** – a MAC computed over the transaction details.

CDA cards additionally compute a **digital signature** over the transaction details and the TC.

The Cambridge Attack

- At Oakland '10 the following **Wedge Attack** was presented, it allows an attacker to make transactions without the card's PIN.
- The wedge manipulates the communication between the card and the terminal so that the terminal believes PIN verification was successful, while the card thinks that a paper signature was used instead.



The Cambridge Attack

- At Oakland ’10 the following **Wedge Attack** was presented, it allows an attacker to make transactions without the card’s PIN.
- The wedge manipulates the communication between the card and the terminal so that the terminal believes PIN verification was successful, while the card thinks that a paper signature was used instead.
- The card’s view of the cardholder verification is transmitted to the terminal in a format which it may not comprehend, and the attack can go undetected even during **online** and **CDA** transactions.
- The attack can easily be prevented, by ensuring that the terminal inspects the card’s view of the cardholder verification.

Our Contribution

- The EMV standard allows the **same RSA key-pair** to be used for both encryption and signature.
- Folklore dictates key separation, but sharing keys reduces processing and storage costs.
- No formal analysis exists that shows whether this is detrimental for the security of EMV or not.
- This is exactly the aim of our paper, we present an attack that exploits key reuse in EMV, together with positive results about upcoming versions of the standards.

A New Attack on EMV

- Our attack exploits the reuse of RSA keys in an EMV card to allow an attacker to make transactions **without** the card's PIN.
- The attack is only applicable to a **CDA** card in an **offline** transaction.
- If the countermeasure against the Cambridge attack is in place our attack would still work!
- The attack builds on Bleichenbacher's attack against RSA with PKCS#1 encoding (CRYPTO '98).

The Bleichenbacher Attack

- PKCS#1 v1.5 specified that the plaintext be encoded as:

$$m = \mathbf{00} \,||\, \mathbf{02} \,||\, \mathbf{Padding\ String} \,||\, \mathbf{00} \,||\, \mathbf{Data}$$

- Assume access to a ciphertext-validity oracle **Valid**$(\cdot)$.
- If **Valid**(c) then $2B \leq m < 3B$, where $B = 2^{8(k-2)}$.
- Using the multiplicative homomorphism of RSA, it is possible to construct a sequence of related ciphertexts such that:
  a. Each ciphertext is valid with probability one half.
  b. Each valid ciphertext found, narrows down the range by half.
- For a 1024-bit RSA modulus, roughly a **million** oracle queries are required to recover m (due to setup overheads).

The Bleichenbacher Attack

- PKCS#1 v1.5 specified that the plaintext be encoded as:

$$m = \mathbf{00} \,||\, \mathbf{02} \,||\, \mathbf{Padding\ String} \,||\, \mathbf{00} \,||\, \mathbf{Data}$$

- Assume access to a ciphertext-validity oracle **Valid**$(\cdot)$.
- If **Valid**(c) then $2B \leq m < 3B$, where $B = 2^{8(k-2)}$.
- Using the multiplicative homomorphism of RSA, it is possible to construct a sequence of related ciphertexts such that:
  a. Each ciphertext is valid with probability one half.
  b. Each valid ciphertext found, narrows down the range by half.
- For a 1024-bit RSA modulus, roughly a **million** oracle queries are required to recover m (due to setup overheads).

The Bleichenbacher Attack

- PKCS#1 v1.5 specified that the plaintext be encoded as:

$$m = \mathbf{00} \,||\, \mathbf{02} \,||\, \mathbf{Padding\ String} \,||\, \mathbf{00} \,||\, \mathbf{Data}$$

- Assume access to a ciphertext-validity oracle **Valid**$(\cdot)$.
- If **Valid**(c) then $2B \leq m < 3B$, where $B = 2^{8(k-2)}$.
- Using the multiplicative homomorphism of RSA, it is possible to construct a sequence of related ciphertexts such that:
  a. Each ciphertext is valid with probability one half.
  b. Each valid ciphertext found, narrows down the range by half.
- For a 1024-bit RSA modulus, roughly a **million** oracle queries are required to recover m (due to setup overheads).

PIN Encryption in EMV

- The encoding used in EMV for PIN is encryption is as follows:

  7F || PIN Block || ICC Challenge || Random Padding

  where the PIN block and the ICC Challenge are 8 bytes long.

- Upon decryption the card performs 3 checks:
  a. Is the ICC Challenge equal to the one it produced?
  b. Is the Header byte equal to '7F'?
  c. Does the PIN in the PIN Block match the one stored in the card?

- If test **b** is carried out first, and its success or failure can be distinguished **(e.g. Timing or Power Analysis)**, then a Bleichenbacher-style attack is possible.

PIN Encryption in EMV

- The encoding used in EMV for PIN is encryption is as follows:

  7F || PIN Block || ICC Challenge || Random Padding

  where the PIN block and the ICC Challenge are 8 bytes long.

- Upon decryption the card performs 3 checks:
  a. Is the ICC Challenge equal to the one it produced?
  b. Is the Header byte equal to '7F'?
  c. Does the PIN in the PIN Block match the one stored in the card?

- If test **b** is carried out first, and its success or failure can be distinguished **(e.g. Timing or Power Analysis)**, then a Bleichenbacher-style attack is possible.

PIN Encryption in EMV

- The encoding used in EMV for PIN is encryption is as follows:

  7F || PIN Block || ICC Challenge || Random Padding

  where the PIN block and the ICC Challenge are 8 bytes long.

- Upon decryption the card performs 3 checks:
  a. Is the ICC Challenge equal to the one it produced?
  b. Is the Header byte equal to '7F'?
  c. Does the PIN in the PIN Block match the one stored in the card?

- If test **b** is carried out first, and its success or failure can be distinguished **(e.g. Timing or Power Analysis)**, then a Bleichenbacher-style attack is possible.

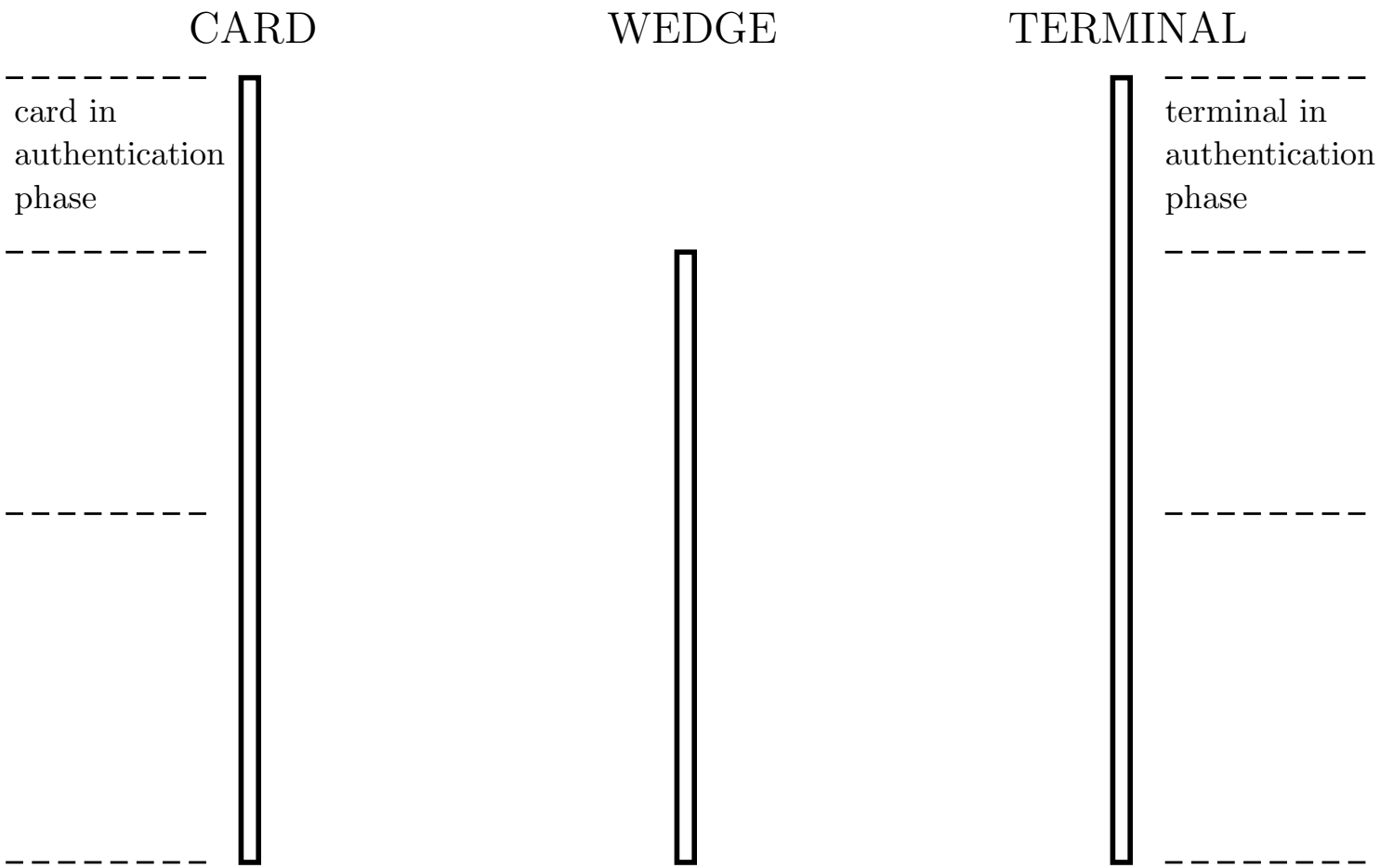
Bleichenbacher's Attack in EMV

- View Bleichenbacher's attack as a black box, which when given a valid ciphertext c and access to a ciphertext-validity oracle recovers the underlying (encoded) message m.
- Alternatively we can view m as the signature of some message whose **encoding** is c, since $m = c^d \bmod N$.
- Thus when a single key pair is used, Bleichenbacher's attack allows us to sign messages whose encodings happen to be also valid ciphertexts.
- In order to sign an arbitrary encoded message μ, we blind it with an integer ρ such that $\rho^e \mu$ is a valid ciphertext.

$$\text{Sign}(\mu) = \rho^{-1} c^d \bmod N$$

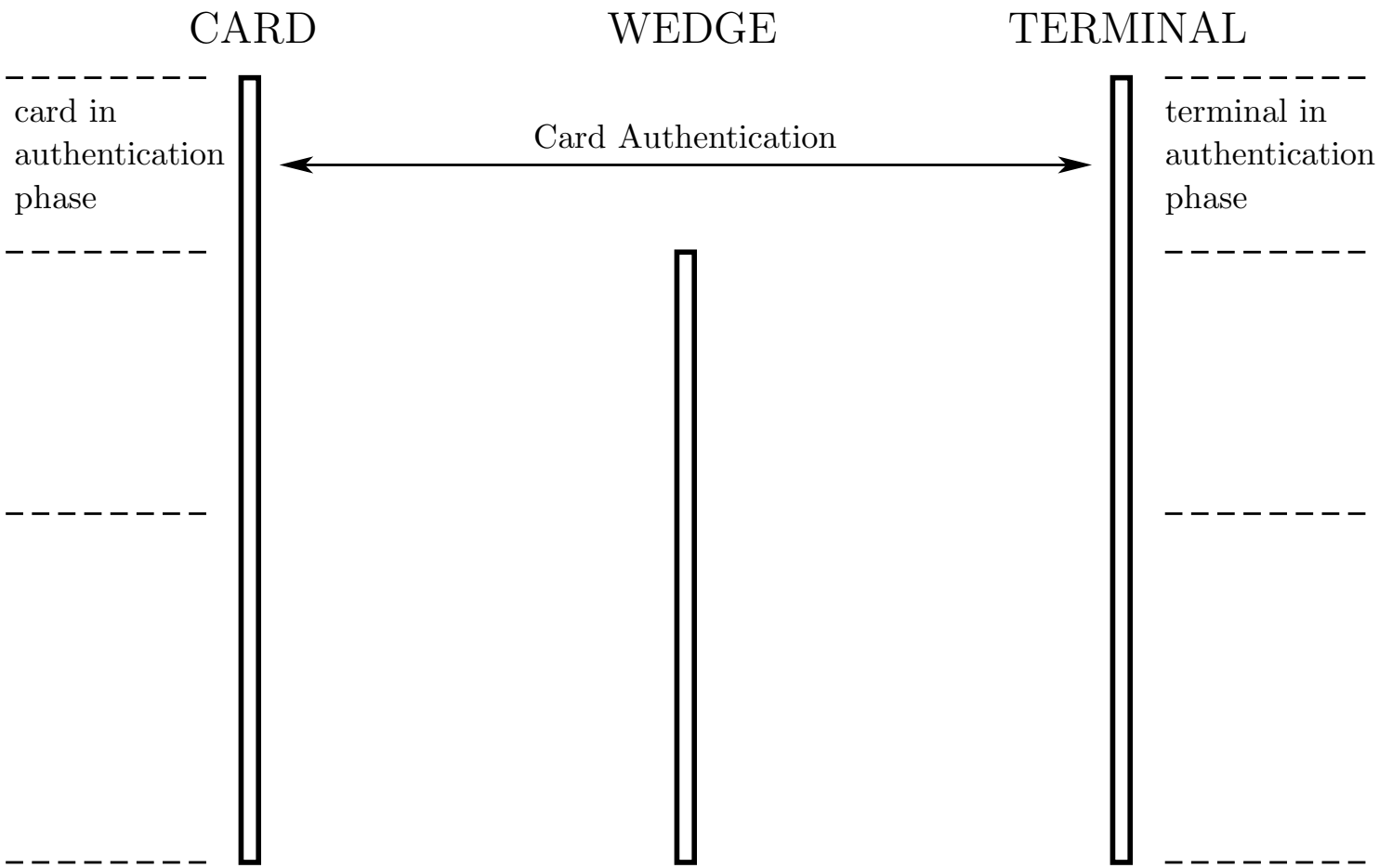
The Attack on a CDA Transaction

Royal Holloway
University of London
Information Security Group



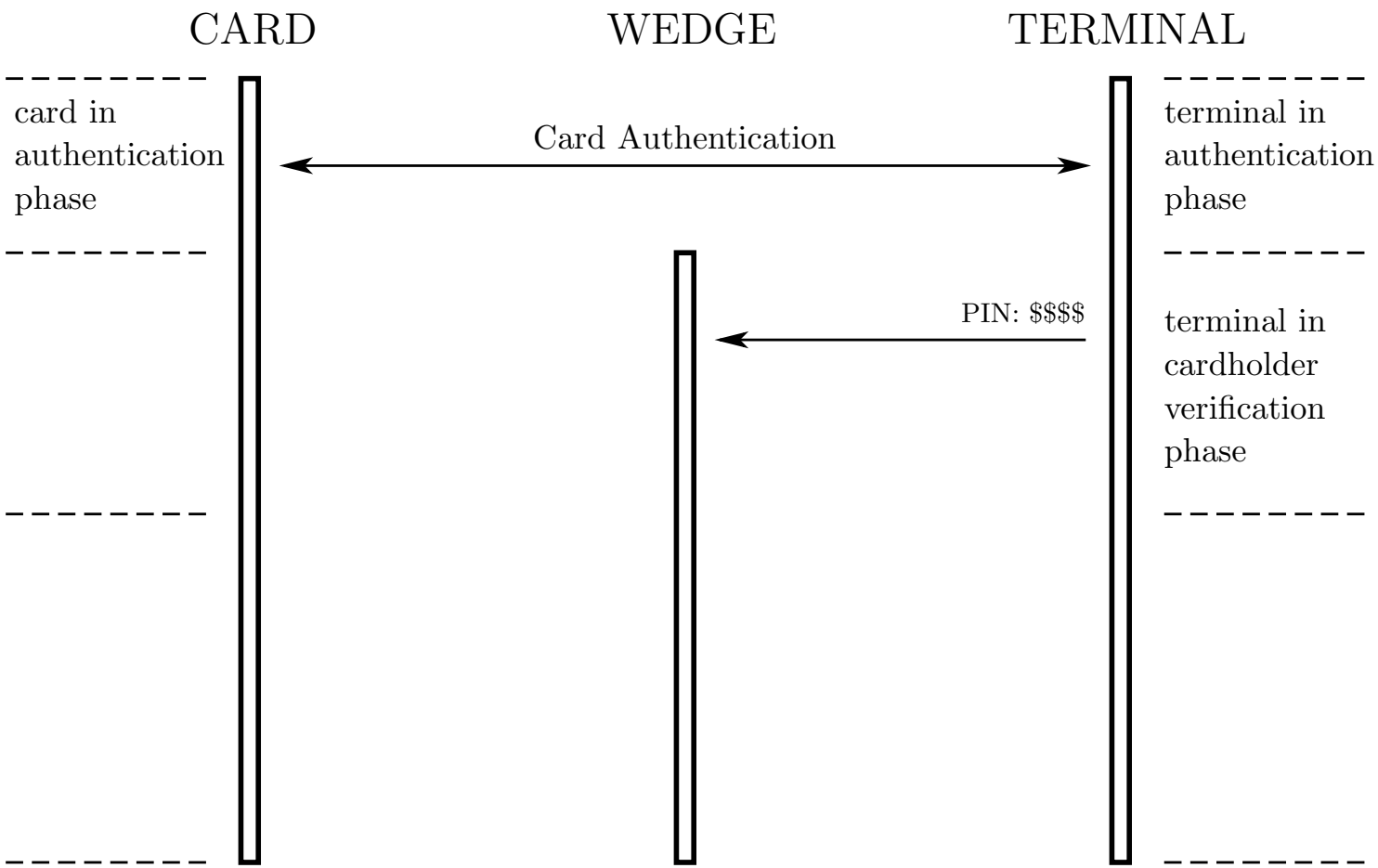
The Attack on a CDA Transaction

Royal Holloway
University of London
Information Security Group



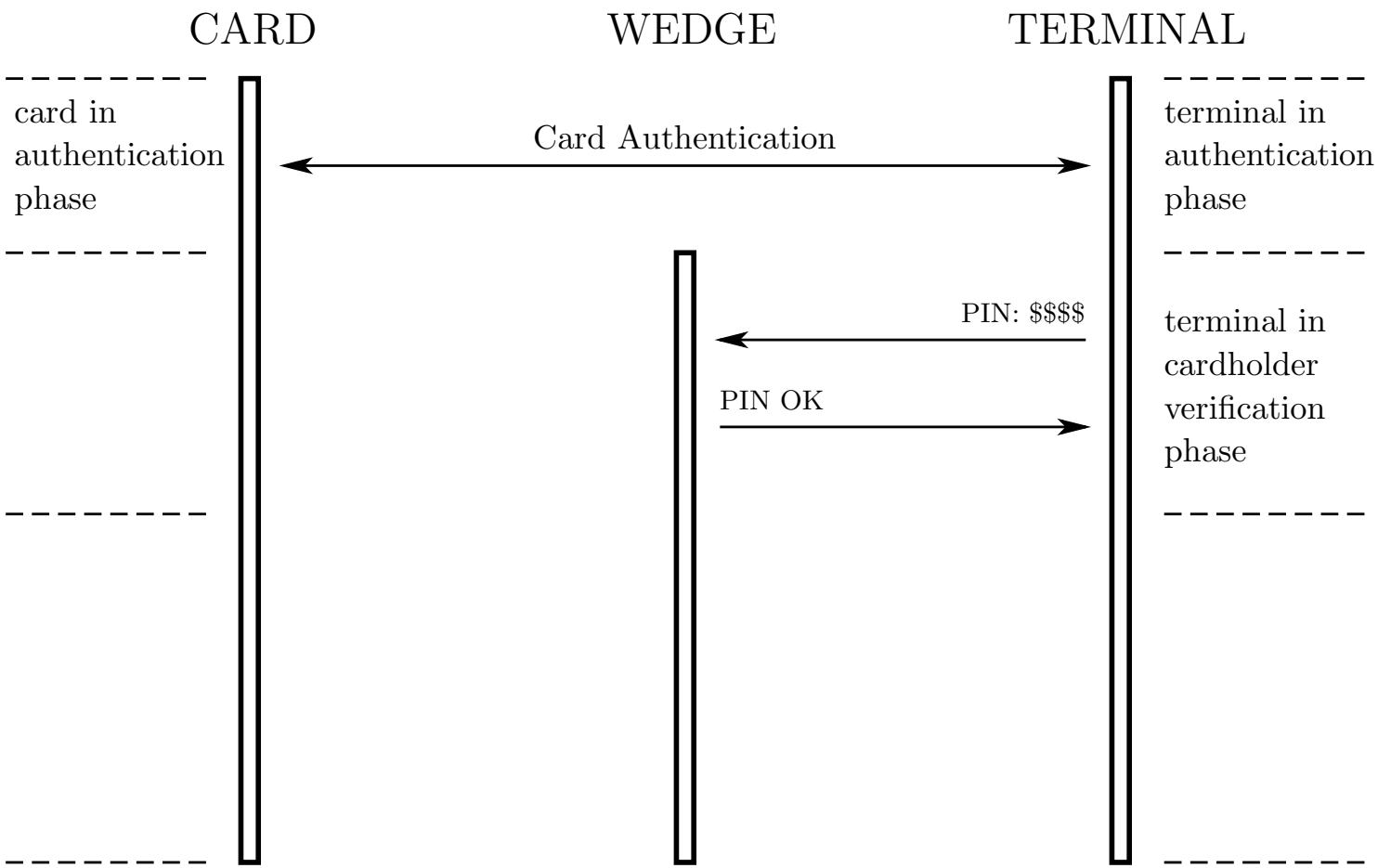
The Attack on a CDA Transaction

Royal Holloway
University of London
Information Security Group



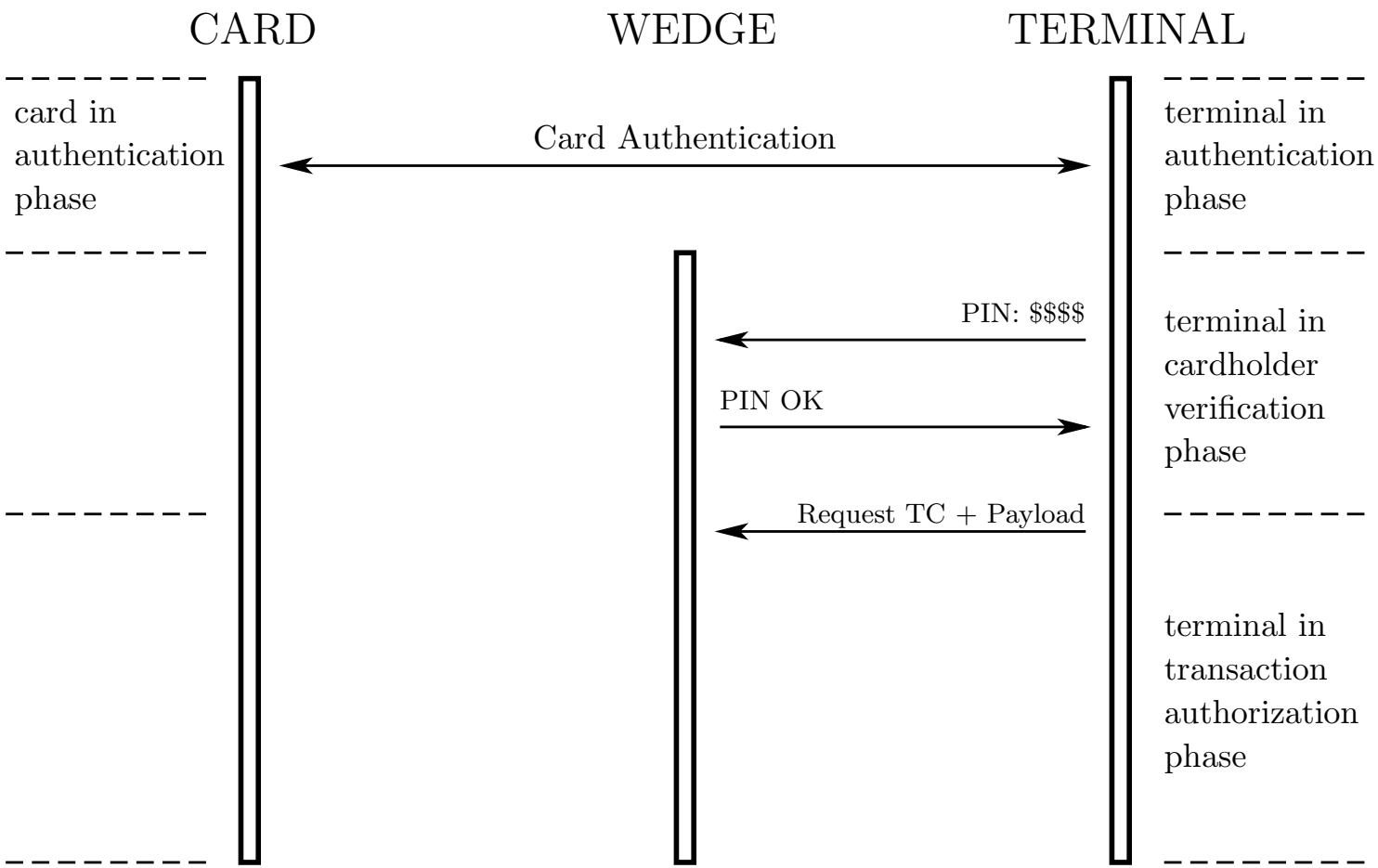
The Attack on a CDA Transaction

Royal Holloway
University of London
Information Security Group



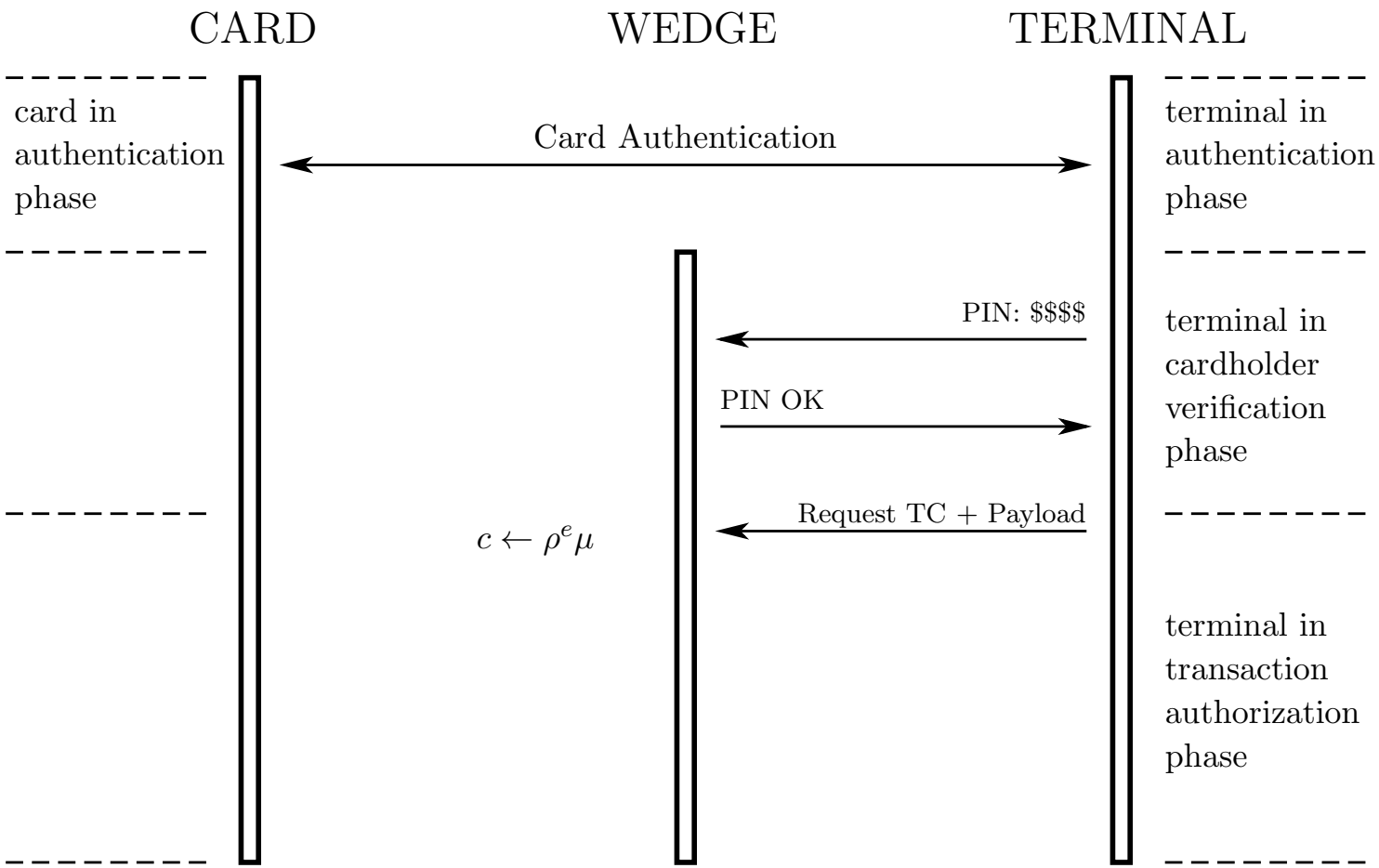
The Attack on a CDA Transaction

Royal Holloway
University of London
Information Security Group



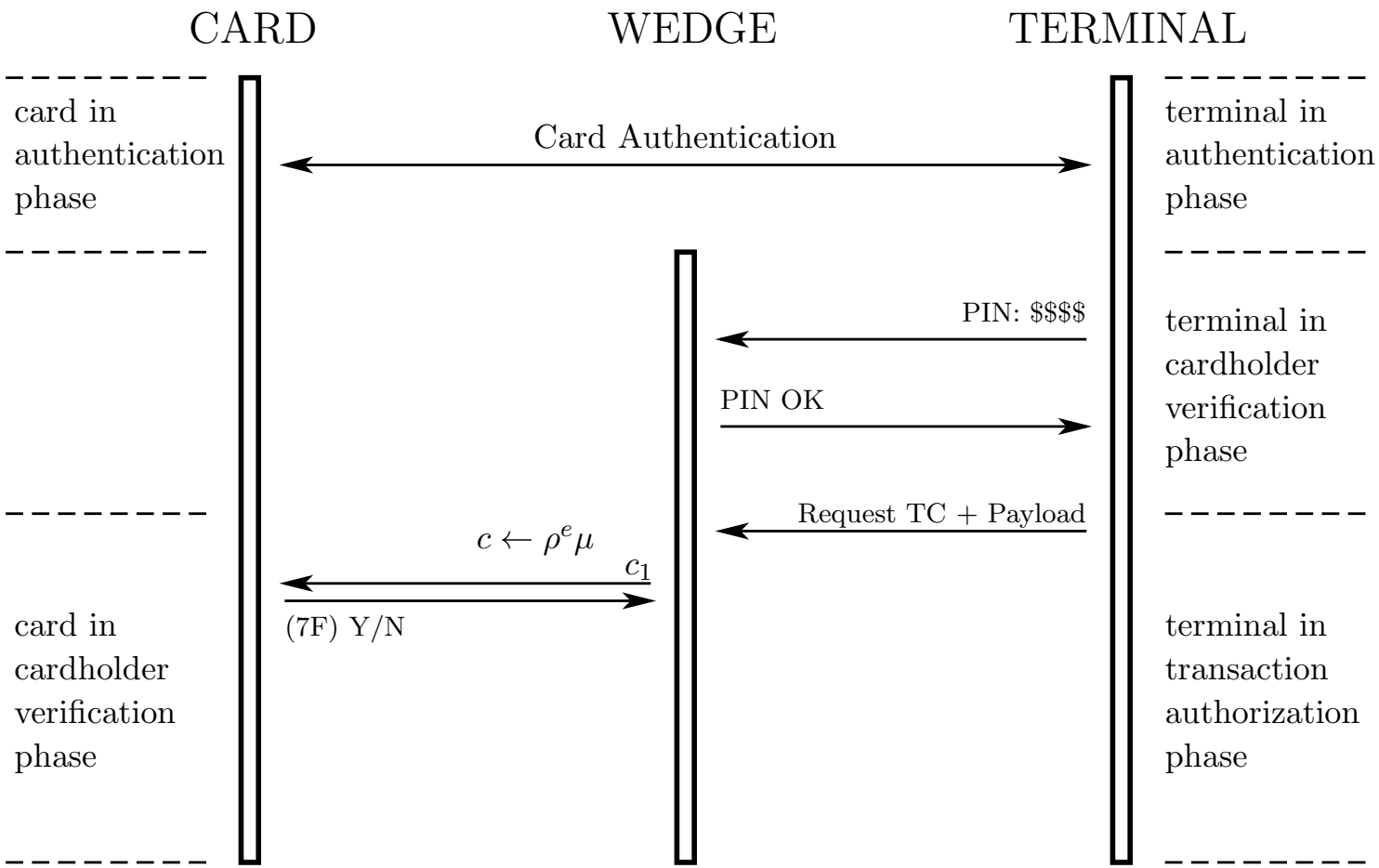
The Attack on a CDA Transaction

Royal Holloway
University of London
Information Security Group



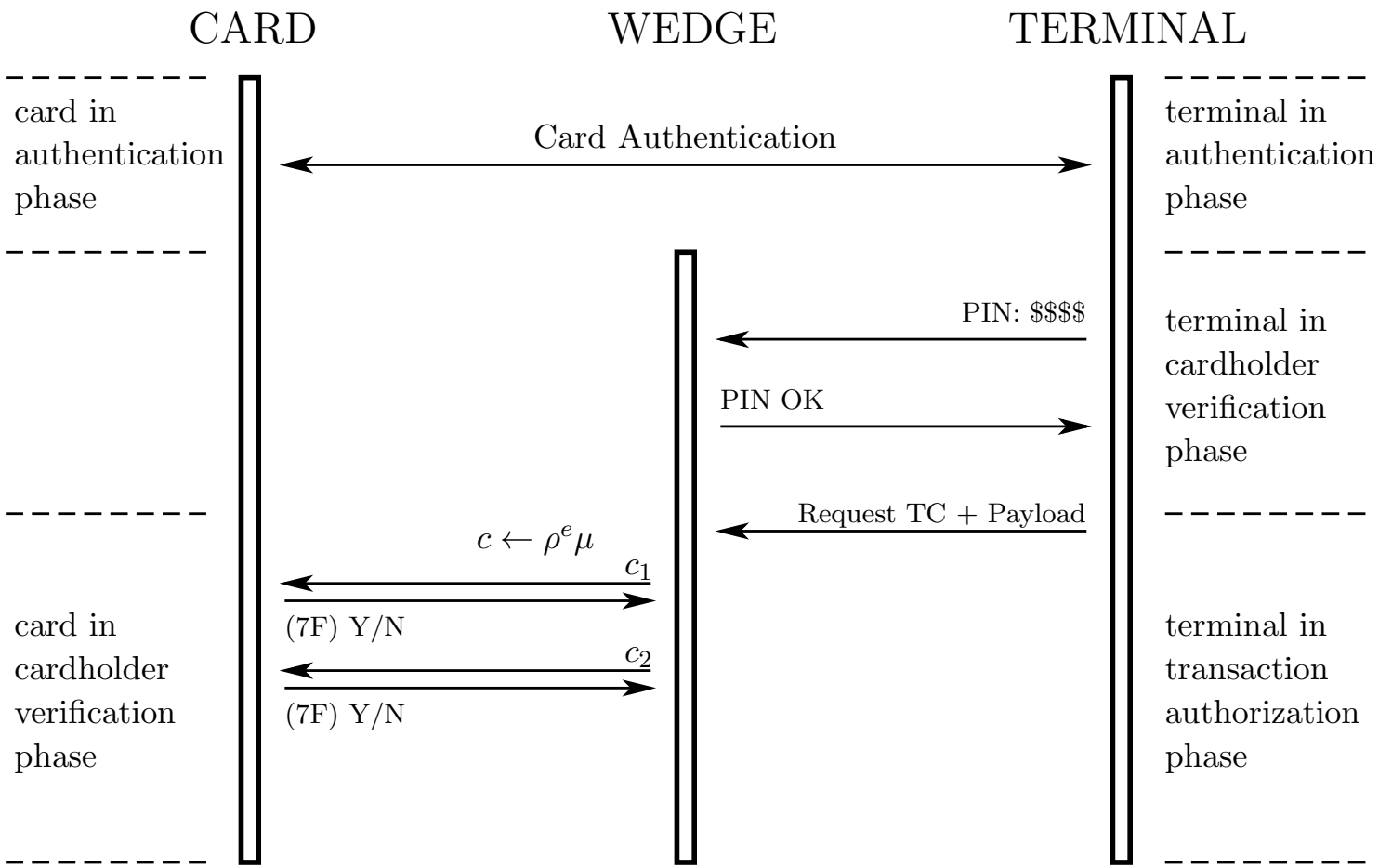
The Attack on a CDA Transaction

Royal Holloway
University of London
Information Security Group



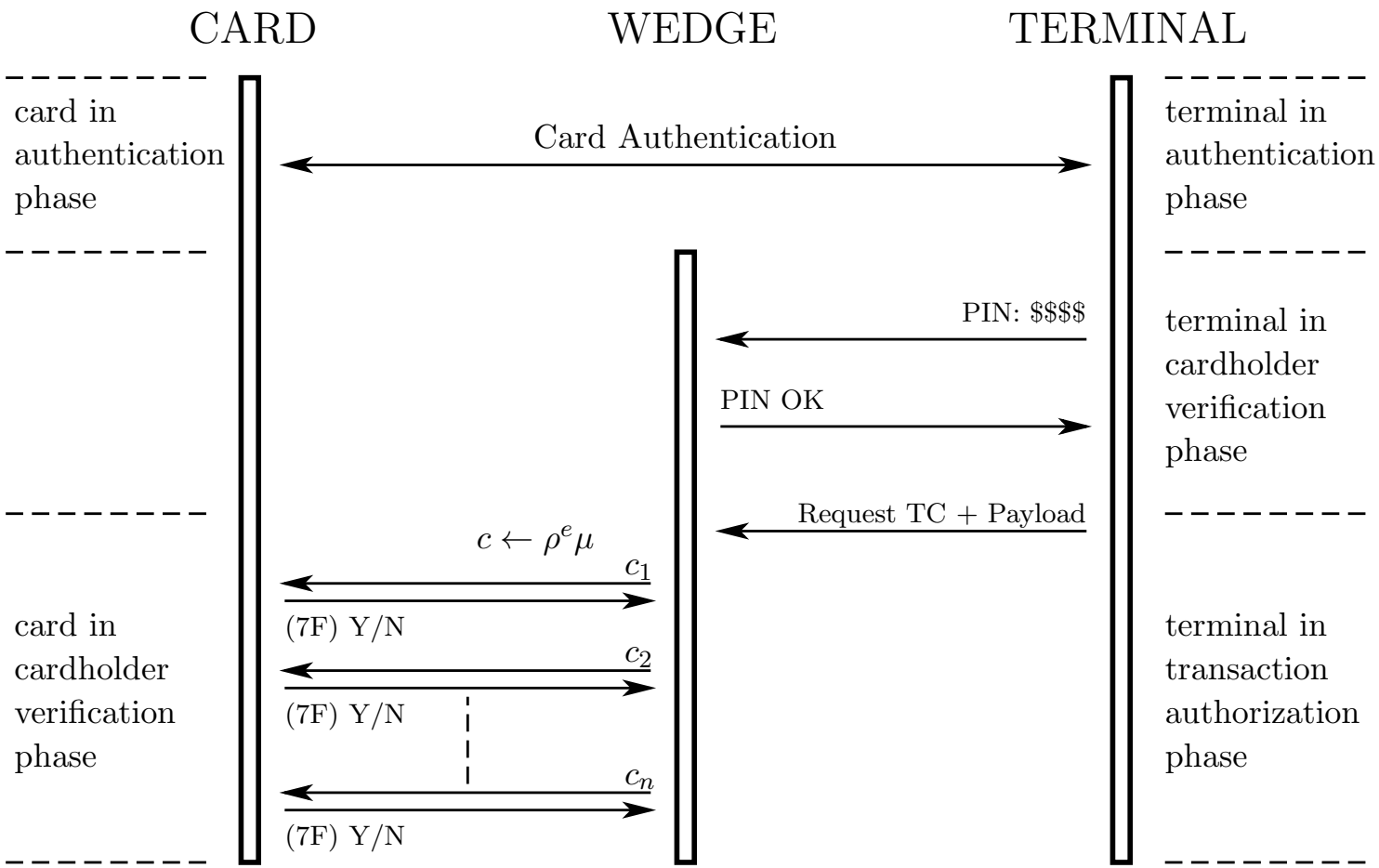
The Attack on a CDA Transaction

Royal Holloway
University of London
Information Security Group



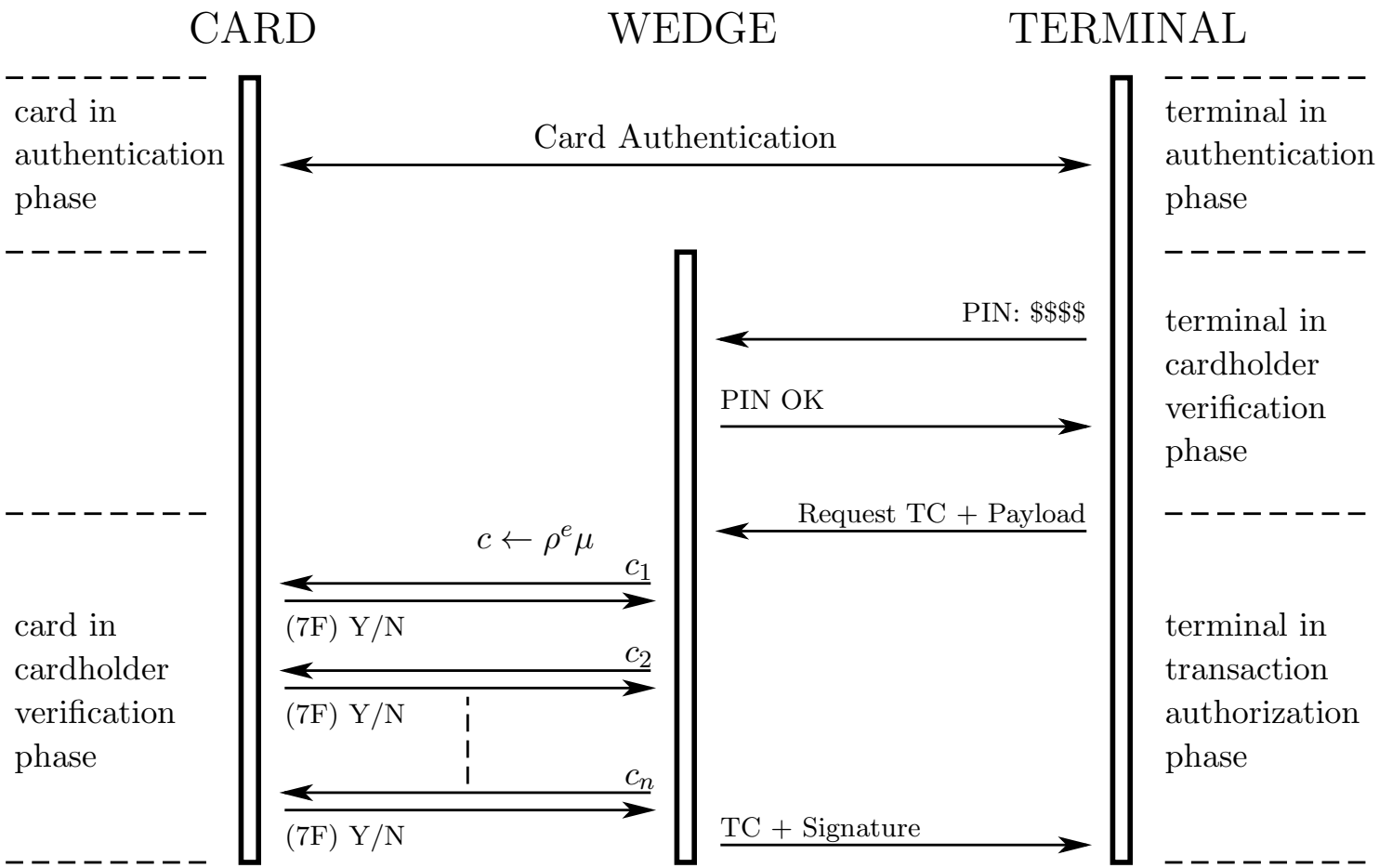
The Attack on a CDA Transaction

Royal Holloway
University of London
Information Security Group



The Attack on a CDA Transaction

Royal Holloway
University of London
Information Security Group



Practical Considerations

- We stress that we did not implement the attack in practice.
- Because the header is only 1 byte long, for a 1024-bit RSA modulus we need roughly **2000** queries to forge a signature.
- EMV cards may maintain both a **PIN try** counter and a **decryption failure** counter. Our attack would not affect the PIN try counter. In the EMV CPA specification the latter is specified to be a 2-byte counter.
- Other factors such as transaction time-outs and the inability to reproduce the '7F' oracle may limit the practicality of our attack.

On the Positive Side

- EMV Co is considering to adopt elliptic curve based algorithms in future versions of the EMV standards.
- More specifically, to use:
  - ECIES (ISO/IEC 18033-2) for PIN encryption.
  - EC-DSA or EC-Schnorr (ISO/IEC 14888-3:2006) to compute digital signatures.
- We show that the two resulting configurations are **jointly secure**, meaning that the security of the individual constituent schemes still holds when they share the same key pair.

Joint Security

- We define a **combined scheme**:

$$(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{KEM.Enc}, \mathsf{KEM.Dec})$$

- EUF-CMA security is augmented by giving the adversary additional access to a decapsulation oracle.
- Similarly IND-CCA security is extended by giving the adversary additional access to a signing oracle.
- A combined scheme is jointly secure if it is **both** EUF-CMA secure in the presence of a decapsulation oracle, and IND-CCA secure in the presence of a signing oracle.

ECIES + EC-Schnorr

In the Random Oracle Model:

Result	Scheme	Security	Assumptions
1	ECIES-KEM	IND-gCCA	gap-DH
2	EC-Schnorr	EUF-CMA	DLP
New	Combined Scheme	Joint Security	gap-DH, gap-DLP

[1] Abdalla, Bellare and Rogaway. *CT-RSA 2001*

[2] Pointcheval and Stern. *J. Cryptology 2000*

ECIES + EC-DSA

Assuming the group is ideal (Generic Group Model):

Result	Scheme	Security	Assumptions
3	ECIES-KEM	IND-CCA	DDH, KDF†
4	EC-DSA	EUF-CMA	$f_{conv}{}^{\ddagger}$, Hash†§
New	Combined Scheme	Joint Security	DDH, $f_{conv}{}^{\ddagger}$, Hash†§

[3] Smart. *Coding and Cryptography 2001*

[4] Brown. *Advances in Elliptic Curve Cryptography 2005*

†Uniform
‡Almost Invertible
§Collision Resistant and Zero-Finder Resistant

Conclusions

- Our attack illustrates the problems in reusing the same key-pair for encryption and signature in the current EMV standards.
- We show that the security of the individual EC-based schemes extends to the **joint setting** under the same assumptions.
- Thus for the elliptic curve based schemes under consideration, one can ‘reuse keys’ and gain substantial efficiency benefits while retaining a similar security margin.

New Constructions of Efficient Simulation-Sound Commitments Using Encryption and Their Applications

Eiichiro Fujisaki

NTT Information Sharing Platform Laboratories

Session ID: **CRYP-202**

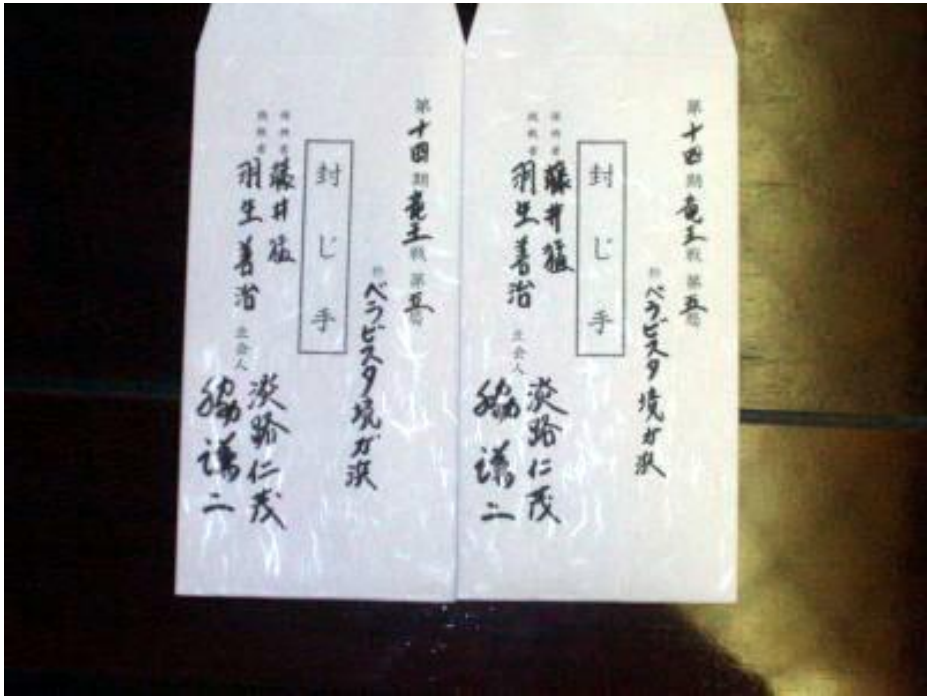
Session Classification: Advanced

Quick Overview

- New frameworks for constructing simulation-sound trap-door commitments (SSTCs)
  - 2-move and 5-move
- Efficient instantiations
  - 2-move assuming CDH in bilinear group.
  - 5-move assuming Factoring.
- What is strong and weak?
  - Strong: ***Tight*** reduction to ***weak (good)*** assumptions.
    - Implies efficient instantiations in the same security level.
  - Weak: Require ***Interactions*** (2-move or 5-move)
    - Previous Works : non-interactive =1-move

Commitments

In a *Shogi* game (a Japanese traditional board game)



Commitments

We focus on commitments in the common reference string model.

CRS: common reference string

Alice

Bob

secret

$x \in \{0,1\}^*$

r :randomness

Commit Phase

$c = Com(x;r)$

Hiding：
Bob does get no information about secret x in the commit phase.

Open Phase

(x,r)

?
$c = Com(x;r)$

Binding：
Alice cannot open c in a different way.

Why we study SSTCs ?

Simulation-sound trap-door commitments are a key ingredient.

- *SSTCs* → cNMo commitments [MY04]
  - cNMo: concurrent non-malleable w.r.t. opening
- Σ-protocols + *SSTCs* → cNM ZK PoKs [Gen04]
  - cNM: concurrent non-malleable
- Ω-protocols + *SSTCs* → UC ZKs [GMY03,MY04]
  - UC: universally composable
- Mix commitments + *SSTCs* → UC commitments [DN02,DG03]
  - Notes: UC commitments → any UC 2-party and multi-party computation.

Agenda

- SSTC =TC +SS binding
  - Trap-door commitment (TC)
  - Simulation-Sound Binding
- Σ-protocols implies TC
- Previous Construction of SSTC
- New frameworks from Encryption (Tag-KEMs)
  - Idea
  - 2 and 5-move Instantiations
- Comparison

Trap-door Commitments

Simulator

secret $x \in \{0,1\}^*$

Trap-door key: tk

w/ (*x*,*r*)

CRS: common reference string

Commit Phase

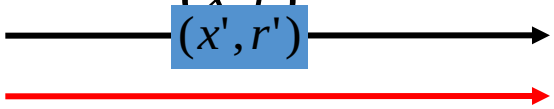
$c = Com(x; r)$

Simulator can open commitment to ***any*** *x'*.

Open Phase

(x, r)

(x', r')



$c = Com(x; r)$

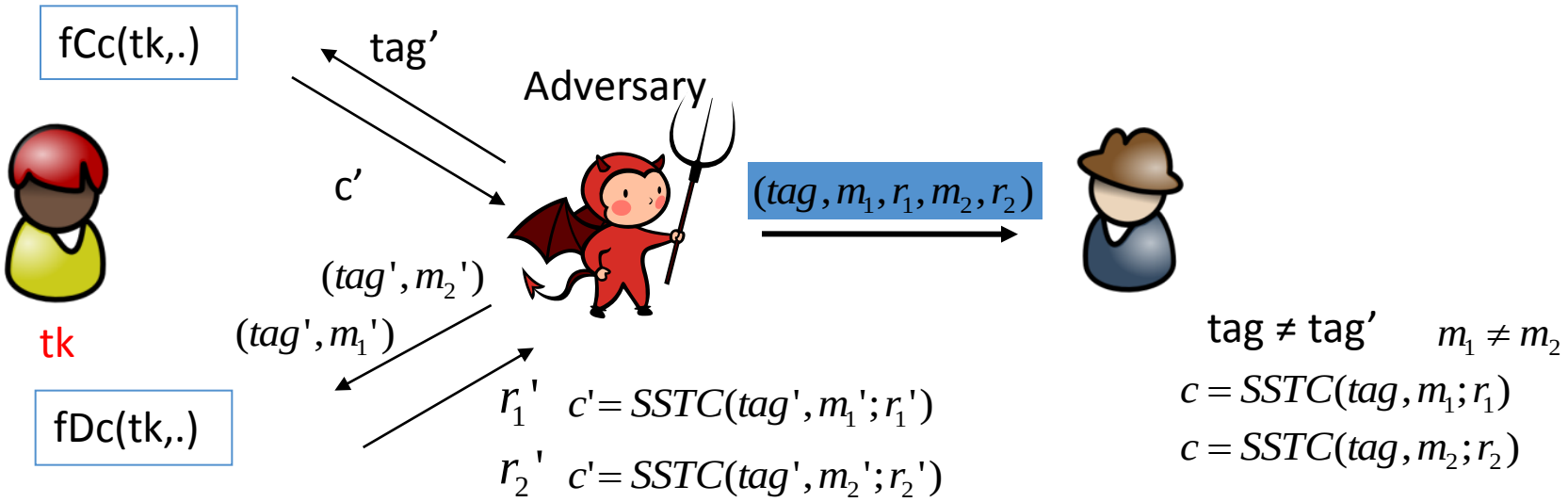
$c = Com(x'; r')$

Ex. Pedersen's Commitment:

$CRS = \{g, h\}$ $tk = \{s\}$ s.t. $h = g^s$ $r' = (x - x')/s + r$ $c = g^x h^r = g^{x'} h^{r'}$

Simulation-Sound (SS) TCs

Simulation-sound binding: Adv is negl. in the following game

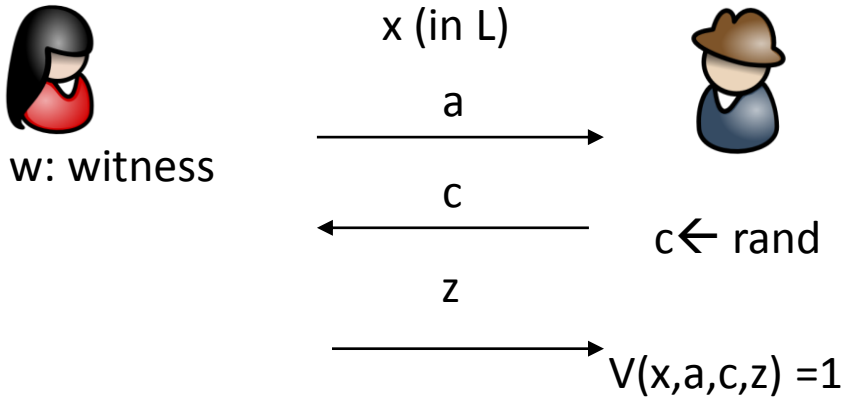


$$\mathsf{Adv}^{\mathsf{ss-bind}}_{A,\mathsf{SSTC}}(n) \triangleq$$

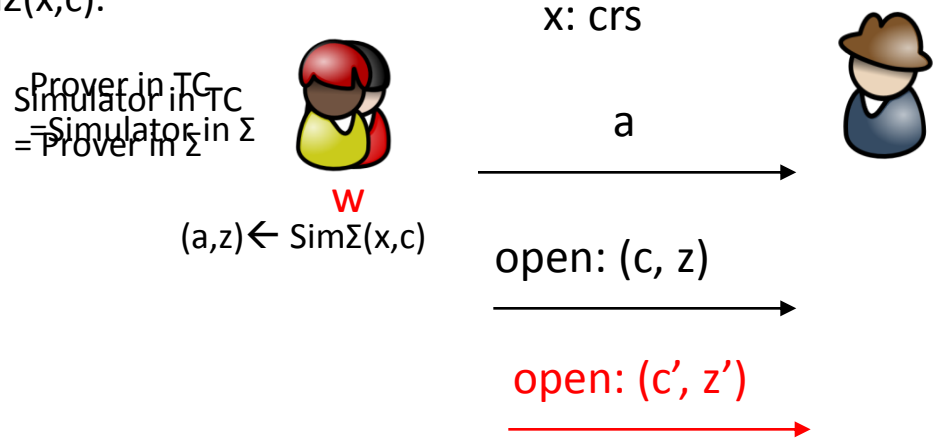
$$\Pr\left[\begin{array}{l}(pk, tk) \leftarrow \mathsf{TKGen}(1^n);\\ (tag, m_1, m_2, r_1, r_2, c) \leftarrow A^{\mathsf{fCc}_{tk}, \mathsf{fTDc}_{tk}}(pk) :\\ c = \mathsf{SSTC}(pk, tag, m_1, r_1) = \mathsf{SSTC}(pk, tag, m_2, r_2)\\ \wedge\ (m_1 \neq m_2)\ \wedge\ tag \notin Q\end{array}\right],$$

Σ-protocol implies TC [FS89,90]

- Σ-protocol on language L.
  - x: an instance in L; w: a witness of x.
  - 3-move public-coin HVZK
  - Completeness
  - Special soundness
  - Special honest verifier ZK
    - (a,z) ← SimΣ(x,c)

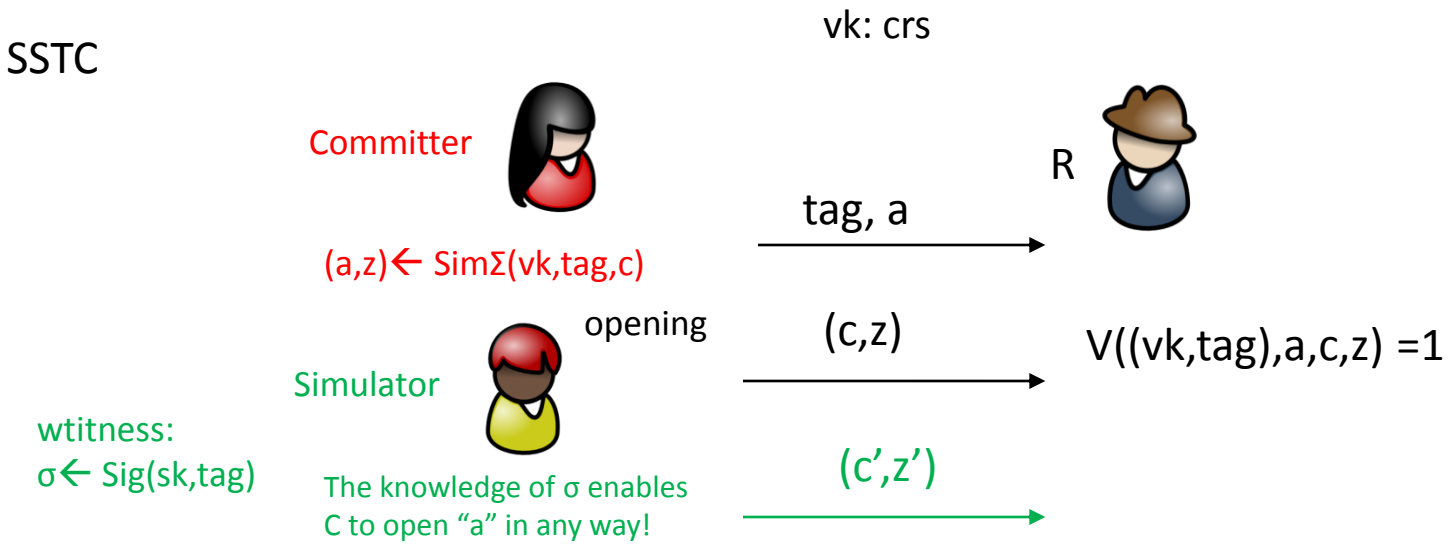


- Trap-door commitment (TC) derived from Σ-protocol on L
  - x (in L): common reference string.
  - c: message (a challenge in Σ)
  - a: commitment to c, where (a,z)←SimΣ(x,c).



Previous framework for SSTC [MY04]

- Assume a Σ-protocol such that the prover knows signature σ on “tag”.
- Commit Phase:
  - Committer: Running the simulator instead of the real Σ protocol. Then send the first message “a” of the simulator. Note that he does not know σ; Hence, he commits to challenge “c” .
  - Simulator (with σ) : Running the real Σ-protocol such that he knows signature σ. Then send the first message “a” of the Σ-protocol.
- Open Phase:
  - Committer: Send (c,z).
  - Simulator (with σ): Open “a” to any value c’ with z’ by using witness σ.

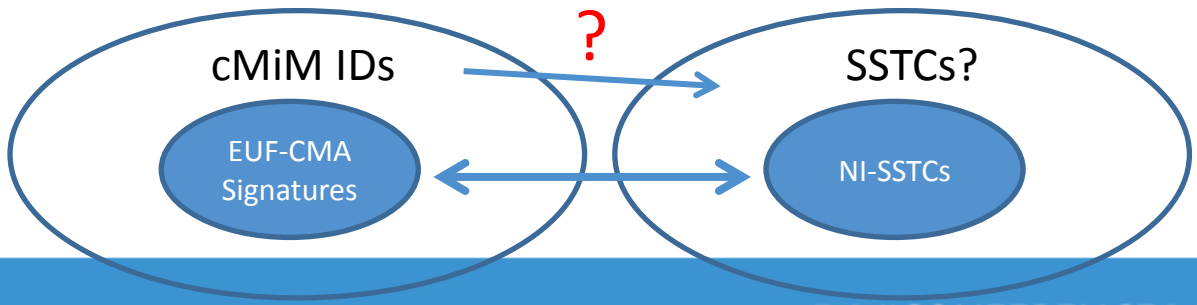


Previous Work (SSTC)

- Using the same framework --- running the simulator of Σ-protocol such that a committer knows a EUF-CMA signature on tag.
  - GMY03: DSA sig. / DSA assumption
  - MY04, Groth03: Cramer-Shoup sig / strong RSA assumption
  - Gen04: BB short sig. / qSDH assumption.
  - DSW08: Waters sig. / CDH assumption
  - NFT10: HW'09 sig. / RSA assumption
- Weakness:
  - The previous schemes have at least one of the following weakness: Strong assumption, loose reduction, or lack of efficiency
    - Q: The weakness mainly comes from the weakness of digital signatures. So, what's if starting with Waters dual-system based signatures based on DLIN with a tight reduction ?
    - A: It depends on whether the dual-system signature has an efficient Σ-protocol. Still, the resulting scheme has at least 7 group elements! Not so practical

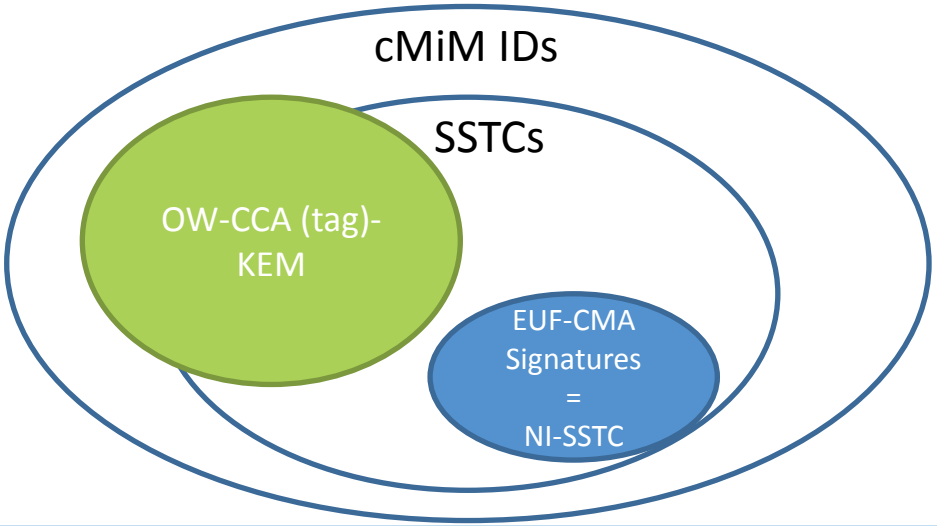
Consider More Efficient Constructions

- Forget non-interactive (NI) SSTCs
  - EUF-CMA signatures imply NI-SSTCs and *vice versa.*
    - Therefore, constructing an efficient NI SSTC is at least as difficult as constructing an efficient EUF-CMA signature scheme.
- Can we bypass signature schemes ?
  - Observation: EUF-CMA sigs imply cMiM IDs.
  - *So, what if starting with (interactive) cMiM identifications?*



Relation between cMiM IDs and SSTCs

- By observation, SSTCs → cMiM IDs
  - The opposite direction (cMiM IDs → SSTCs) is not known. Maybe false.
- By observation, OW-CCA PKE (or tag-KEM) → cMiM IDs
  - Which paper mentioned it first ? Implicitly, [DDN91]? Explicitly, at least [BFGM01], [AA11] and this work.
- This Work: OW-CCA (tag) KEM + *some conditions* → SSTCs



cMiM secure ID from OW-ftCCA Tag-KEM

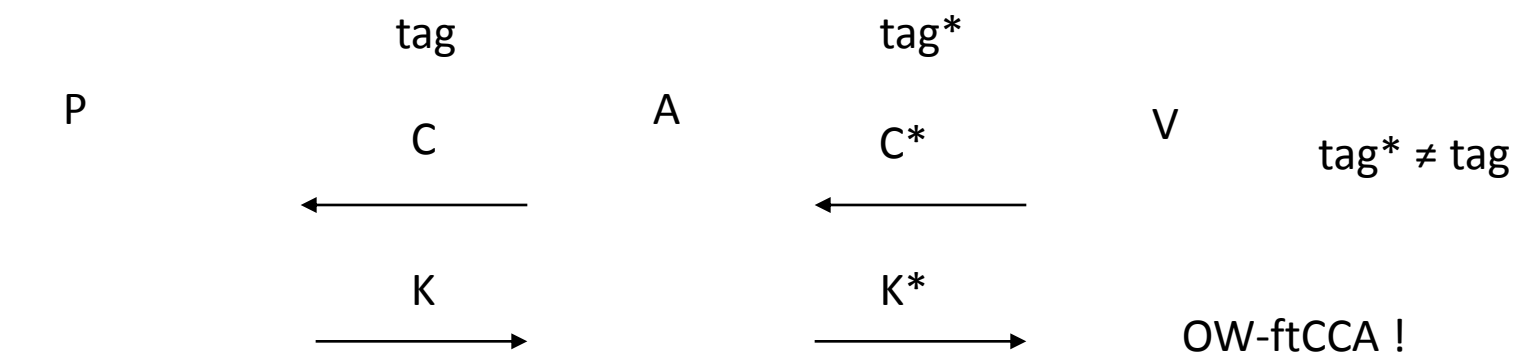
$P(pk, sk, tag)$ $\qquad$ $V(pk, tag)$

$\xleftarrow{C}$ $\qquad$ $(C, K) \leftarrow \mathsf{Enc}(pk, tag)$

$K' := \mathsf{Dec}(sk, tag, C)$

$\xrightarrow{K'}$ $\qquad$ accepts if and only if $K' = K$

cMiM Attack



Top-level Idea: SSTC from Tag-KEM

pk: crs

Committer

Receiver

C

(C,K) ← Enc(pk,tag)

(a,z) ← SimΣ(pk,tag,C,m)
as if she knows K (but she does not know K).

a

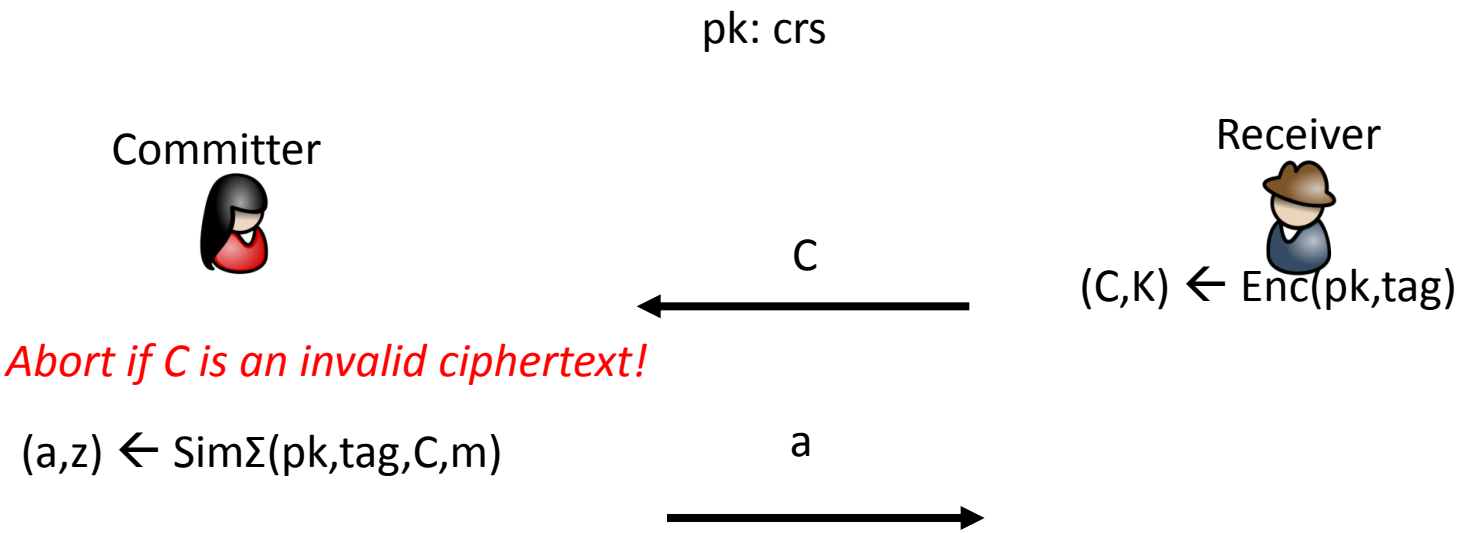
Apparently good, but what what if the receiver sends a fake ciphertext C?
Then, there is no K, which implies that the trap-door property is destroyed!

sk

The simulator extracts K=Dec(sk,tag,C) and opens "a" to any m' with z'!

∀m' , (m',z')

2-move SSTC from publicly-verifiable Tag-KEM



Indeed, such publicly-verifiable Tag-KEMs exist based on CDH assumption in bilnear groups" [Kiltz06,Wee10]

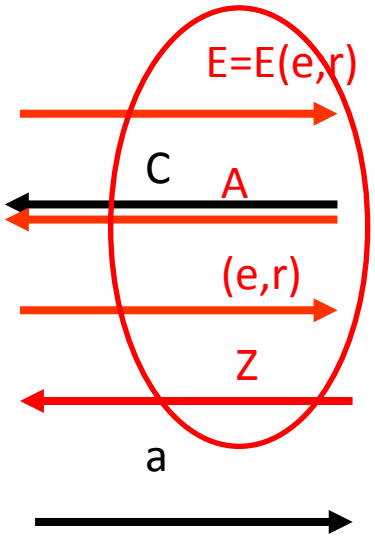
Trial: SSTC from *non-publicly-verifiable* Tag-KEM

pk: crs; public key for Tag-KEM

E: crs; CCA2 Enc

Committer

Receiver



(C,K) ← Enc(pk,tag)

(a,z) ← SimΣ(pk,tag,C,m)

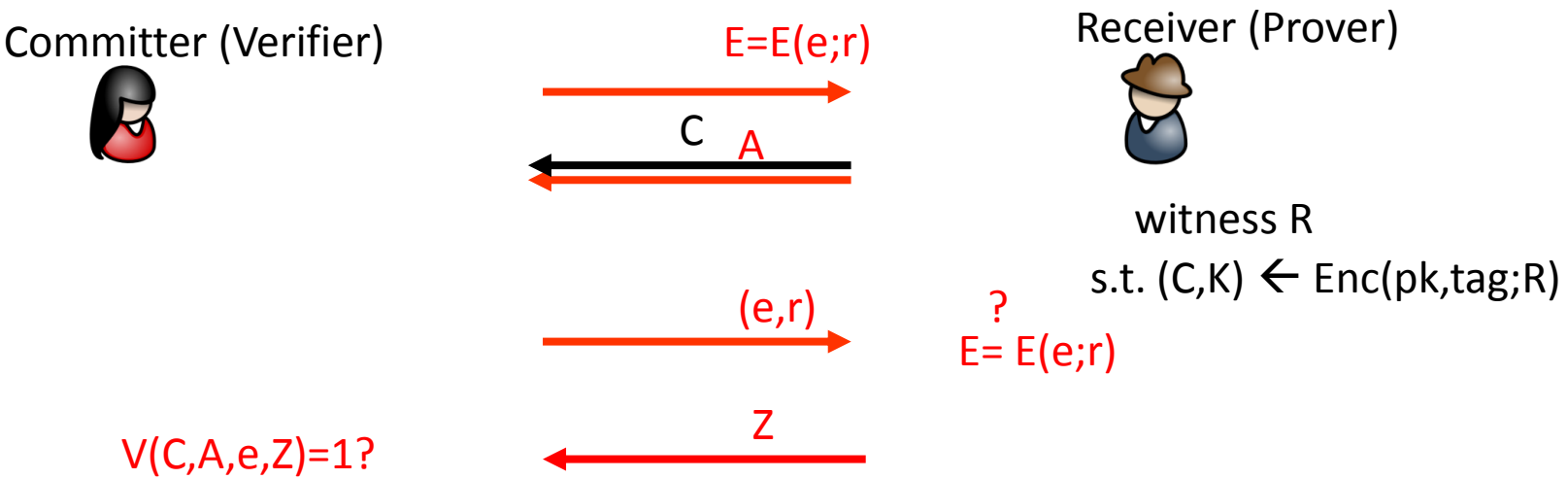
V(C,A,e,Z)=1 iff C is valid.

?
E= E(e;r)

Receiver proves that C is valid.

We need cNM ZK on L={C| C is a valid ciphertext}

pk: crs; public key for Tag-KEM
E:crs; CCA Enc

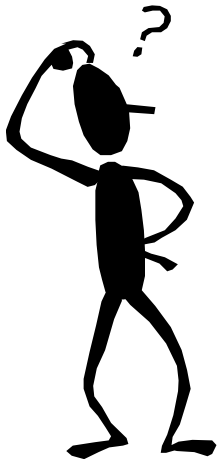


(A,e,Z) is an output of Σ-protocol on common instance C

Concurrent ZKness: OK due to CCA ENC E and Σ-protocol.
Soundness: does not hold for an arbitrary Σ-protocol.

Wait..

We need cNM ZK in order to construct a SSTC, but cNM ZKs (POK) are usually constructed from SSTCs …



We do not need cNMZK *Proof of knowledge,* but cNMZK on language. In addition, we only require cNMZK on a special language such that L ={C | C is a valid ciphertext}.

If Tag-KEM has a special kind of Σ-protocol, denoted *weak extractable Σ-protocol*, then we can prove that the protocol above is cNMZK.

Weak Extractable Sigma Protocols

- Note that in a Σ-protocol, *if x not in L*, the first message of simulation *"a"* is a statistically-binding commitment to *challenge "c"*.
  - Namely, *"c"* is uniquely determined.
- Informally, a weak extractable Σ protocol is a special Σ protocol in the CRS model, where additionally,
  - Every x not in L, every *"a"*, and every *"c"*, one can easily check whether there is *"z"* such that V(crs,*a*,*e*,*z*)=1, if he is given trap-door tk (weak extractability).
- Fortunately, several Tag-KEMs including factoring-based one [HK09] has such a special Σ protocol.

5-move SSTC from Tag-KEM w/ weak extractable Σ-protocol

pk: crs; public key for Tag-KEM

E: crs; CCA2 Enc

Committer

Receiver

E=E(e;r)

C

A

(C,K) ← Enc(pk,tag)

(e,r)

? E= E(e;r)

Receiver proves that C is valid *using a weak extractable Σ protocol.*

(a,z) ← SimΣ(pk,tag,C,m)

Z

V(C,A,e,Z)=1 iff C is valid.

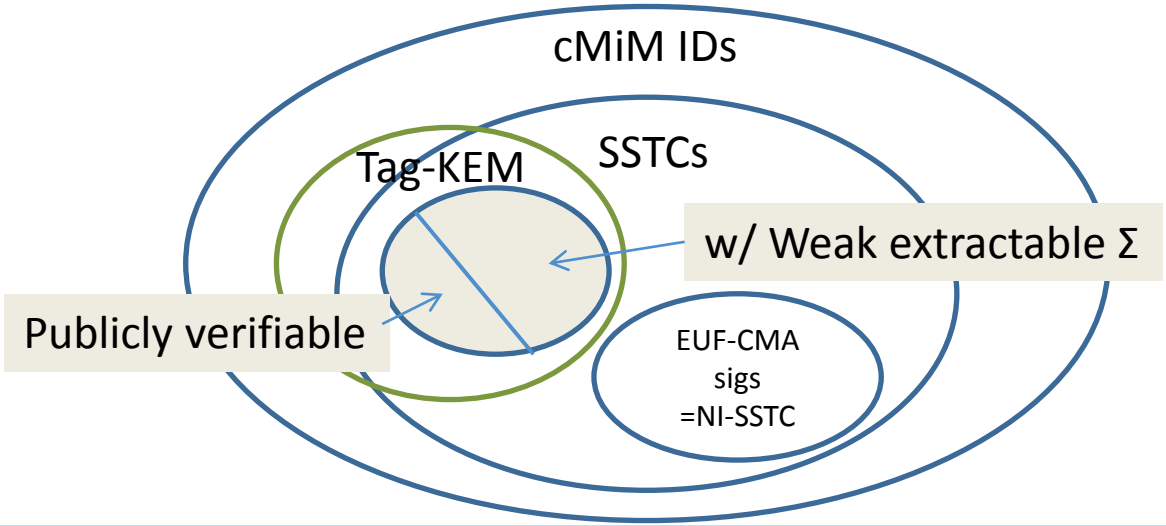
a

∀m' , (m',z')

The simulator can always obtain K from C and open "a" to any m' with z'!

To sumalize..

- Proposed new frameworks for constructing SSTCs using encryption (Tag-KEM).
- Instantiations
  - 2-move if Tag-KEM is *publicly verifiable*
  - 5-move if Tag-KEM has *a weak extractable Σ-protocol.*



Comparison

SSTC schemes	Protocol Efficiency	Assumption	Reduction	Type
GMY 03	Efficient	DSA	------	DSA
MY04/DG 03	Not efficient	sRSA	Tight	Cramer-Shoup sig.
Gen04	Efficient	qSDH	Tight	BB short sig.
DSW08	Efficient but long crs.	CDH	Loose	Waters sig.
NFT10	Inefficient	RSA	Loose	HW sig.
This work (2-move)	Efficient	CDH	Tight	Kiltz's Tag-KEM
This work (5-move)	Efficient	Factoring	Tight	HKTag-KEM

Thank you..