Concurrently-Secure Blind Signatures…

Carmit Hazay
Yehuda Lindell
Bar-Ilan University

Jonathan Katz
Chiu-Yuen Koo
University of Maryland

Blind Signatures [Chaum]

- Enables a user to obtain a signature from a signer on a message, without the signer later being able to "link" this message/signature to this particular user
- Motivation: e-cash, e-voting
  - Useful when values need to be <u>certified</u>, yet <u>anonymity</u> should be preserved

Example: e-cash (simplified)

- Bank has public key PK
- User:
  - Sends its account information to the bank
  - Obtains a signature σ on a "coin" c (e.g., a random string)
- When the user later presents (c, σ) to a merchant, the merchant can verify σ
  - Should be infeasible to trace (c, σ) to a particular user

Example: e-voting (simplified)

- Registration authority has public key PK
- Voter:
  - Proves she is a valid voter…
  - Obtains signature σ_i on public key pk_i (generated and used for this application only)
- Voter can later cast her vote v (on a public bulletin board) by posting (pk_i, σ_i, v, σ)
  - Should be infeasible to trace a public key to a particular voter

Requirements

- Need to consider security requirements of both the signer and the user
  - Protection of signer: **Unforgeability**
  - Protection of user: **Blindness**

Unforgeability

- Intuitively:
  - Like unforgeability for standard signatures (but interaction with user and signer may not unambiguously define a message…)
  - If a malicious user interacts with the (honest) signer n times, it should be infeasible for the user to output valid signatures on $n+1$ distinct messages
- Note: these interactions might be
  - Sequential
  - Parallel
  - Concurrent

Easy to show a protocol secure against *parallel* attacks, but *not* concurrent ones

Blindness

- Intuitively:
  - Malicious signer should be unable to link a (message, signature) pair to any particular execution of the protocol
  - A bit messy to formalize…

Well, sort of…

Blindness ("standard" def'n)

- (Malicious) signer outputs PK, m_0, m_1
- Random bit b selected; signer interacts concurrently with $U(m_b)$, $U(m_{1-b})$
  - If either user-instance aborts, signer gets nothing
  - If neither user-instances aborts, signer gets the messages m_0, m_1 and their signatures
- Pr[Signer guesses b] – ½ = negl

Necessity of dealing with abort

- Say signer can induce a *message-dependent* abort
  - I.e., can act so that user aborts if using m_0 but not if using m_1
- Consider the following attack:
  - Act as above in first session; act honestly in second session
  - Learning whether users abort or not enables correct guess of b
  - Leads to "real-world" attack

Extending blindness def’n

- It is not clear how to extend the “stand-alone” definition to the general case of polynomially-many users
  - Issue is how to deal with aborts…
- In retrospect, not intuitively clear that the “standard” definition provides a good model even in the two-user case

Rethinking the definition

- Let us recall what we want…
  - Signer should be unable to link (m_i, σ_i) with any particular *completed* session, *any better than random guessing*
  - Equivalently (2-user case), want:

Pr[guess b | *both* sessions completed] – ½ = negl

  - Furthermore, in applications the messages are chosen by the *user*, not the signer, and from a known distribution

A new definiti

Could be uniform over two messages

- Signer outputs PK, $\mathcal{D}$
- $\{m_i\}$ chosen according to $\mathcal{D}$
- Signer interacts with $U(m_1), \ldots, U(m_l)$; given (m_i, σ_i) for completed sessions in random permuted order
- Signer *succeeds* if it identifies a message/signature pair with its correct session
- Require: for all p,

$$\Pr[\mathrm{Succ} \wedge \#\mathrm{completed} = p] - 1/p\,(\Pr[\#\mathrm{completed} = p]) < \mathrm{negl}$$

Prior work I

- Blind signatures introduced by Chaum
  - Chaum's construction later proved secure by [BNPS02] based on the "one-more-RSA" assumption in the RO model
- Provably-secure schemes (RO model)
  - [PS96] – logarithmically-many sigs
  - [P97] –poly-many sigs
  - [A01, BNPS02, B03] – concurrently-secure

Prior work II

- Provably-secure schemes (standard model)
  - [JLO97] – using generic secure 2PC
  - [CKW04] – efficient protocol
    - Both give sequential unforgeability only

Prior work III

- [L03] – impossibility of concurrently-secure blind signatures (without setup)!
- Lindell's impossibility result has recently motivated the search for concurrently-secure signatures *in the CRS model*
  - E.g., [O'06, KZ'06, F'06]
  - Circumventing [L03] explicitly mentioned as justification for using a CRS

Is a CRS really necessary…?

- The impossibility result seems to suggest so…
  - …but in fact, [L04] only rules out *simulation-based* security (with black-box reductions)

- Question: can we circumvent the impossibility result by using *game-based* definitions?

Main result

- We show the first concurrently-secure blind signature scheme
  - Standard assumptions, no trusted setup
- Remark: work of [BS05] could seemingly be used as well
  - Would require super-poly hardness assumptions, something we avoid here

Perspective

Impossibility results must be interpreted carefully…

The construction

- Preliminaries
- Fischlin's approach to blind signatures
- A partial solution
  - (Using complexity leveraging)
- The PRS cZK protocol
- The full solution

Preliminaries I

- ZAPs [DN00]
  - 2-round WI proofs for NP; 1^{st} message independent of statement
  - Constructions given in [DN00,BOV03,GOS06a,GOS06b]

Preliminaries II

- Ambiguous commitment (cf. [DN02])
  - Two types of keys
    - One type gives perfect hiding; other type gives perfect binding (with trapdoor for extraction)
  - Easy constructions based on standard number-theoretic assumptions (e.g., DDH)

Fischlin’s approach

- Previous blind signature schemes define a protocol to generate some “standard” signature
  - “Blinding” [Chaum, …]
  - Secure computation approach
- Fischlin takes a different route

Fischlin's approach

CRS: pk, r

<u>Signer(SK)</u> | <u>User(PK, m)</u>

$com = Com(m)$ ←

$\sigma = Sign_{SK}(com)$

σ →

$C = E_{pk}(com \mid \sigma)$
NIZK proof π:
{C correct for m}

Removing the CRS…

- Removing r:
  - Use ZAP instead of NIZK
  - Need to introduce “extra” witness in protocol
  - Use Feige-Shamir trick…
- Removing pk:
  - Want semantic security, yet extraction!
  - Use complexity leveraging…
  - Commitment scheme that is hiding for PPT adversaries, but allows extraction in time T(k)
  - Other components should have T(k)-time security

A partial solution

PK: pk', y_0, y_1, r

Signer | **User(m)**

$\xleftarrow{\text{com} = \text{Com}(m)}$

$\sigma = \text{Sign}_{sk'}(\text{com})$

$\xrightarrow{\sigma}$

WI-PoK: x_0 or x_1 $\Longrightarrow$

$C_1 = \text{Com}^*(\text{com} \mid \sigma)$

$C_2 = \text{Com}^*(0^k)$

ZAP π:

$\{C_1$ correct for $m\}$ or

$\{C_2$ correct for $y_0/y_1\}$

Toward a full solution…

- In our full solution, we use (a modification of) the cZK protocol of [PRS02]
  - Modified to be an argument of knowledge (in stand-alone sense)
  - (Unfortunately…) we cannot use cZK as a "black-box" but instead must use specific properties of the PRS simulation strategy
    - Look-aheads and straight-line simulation
    - Fixed schedule

Main idea

- Instead of using complexity leveraging, use an ambiguous commitment scheme
  - Signer includes commitment key as part of its public key
  - To prevent cheating, signer must give cZK proof that the key is of the correct type

First try

PK: pk′, pk*, r

Signer		User(m)
	com = Com(m) $\longleftarrow$	
$\sigma = \mathrm{Sign}_{sk'}(\mathrm{com})$	$\xrightarrow{\sigma}$	
	cZK: pk* correct $\Longrightarrow$	$C = \mathrm{Com}_{pk*}(\mathrm{com} \mid \sigma)$ ZAP π: {C correct for m} or {pk* correct}

Proof?

- Fairly straightforward to show the following:
  - Given user U who interacts with the (honest) signer and outputs n+1 signatures on distinct messages with non-neg probability…
  - …can construct forger F who interacts with a (standard) signing oracle and outputs n+1 signatures on distinct messages with non-neg probability
- Problem:
  - F might make >n queries (even if U does not)!

Modification

- The signer will append a random nonce to what is being signed
- The forger F we construct will still output n+1 signatures but make >n oracle queries…
  - …but the signatures output by F are (in some sense) *independent* of the nonces used during rewinding
    - With high probability, one of the signatures output by F will be a *forgery*

The protocol

PK: pk′, pk*, r

<u>Signer</u>

<u>User(m)</u>

com = Com(m)

$nc \in \{0,1\}^k$

$\sigma = \mathrm{Sign}_{sk'}(nc|com)$

nc, σ

$C = \mathrm{Com}_{pk*}(nc|com|\sigma)$

ZAP π:

{C correct for m} or

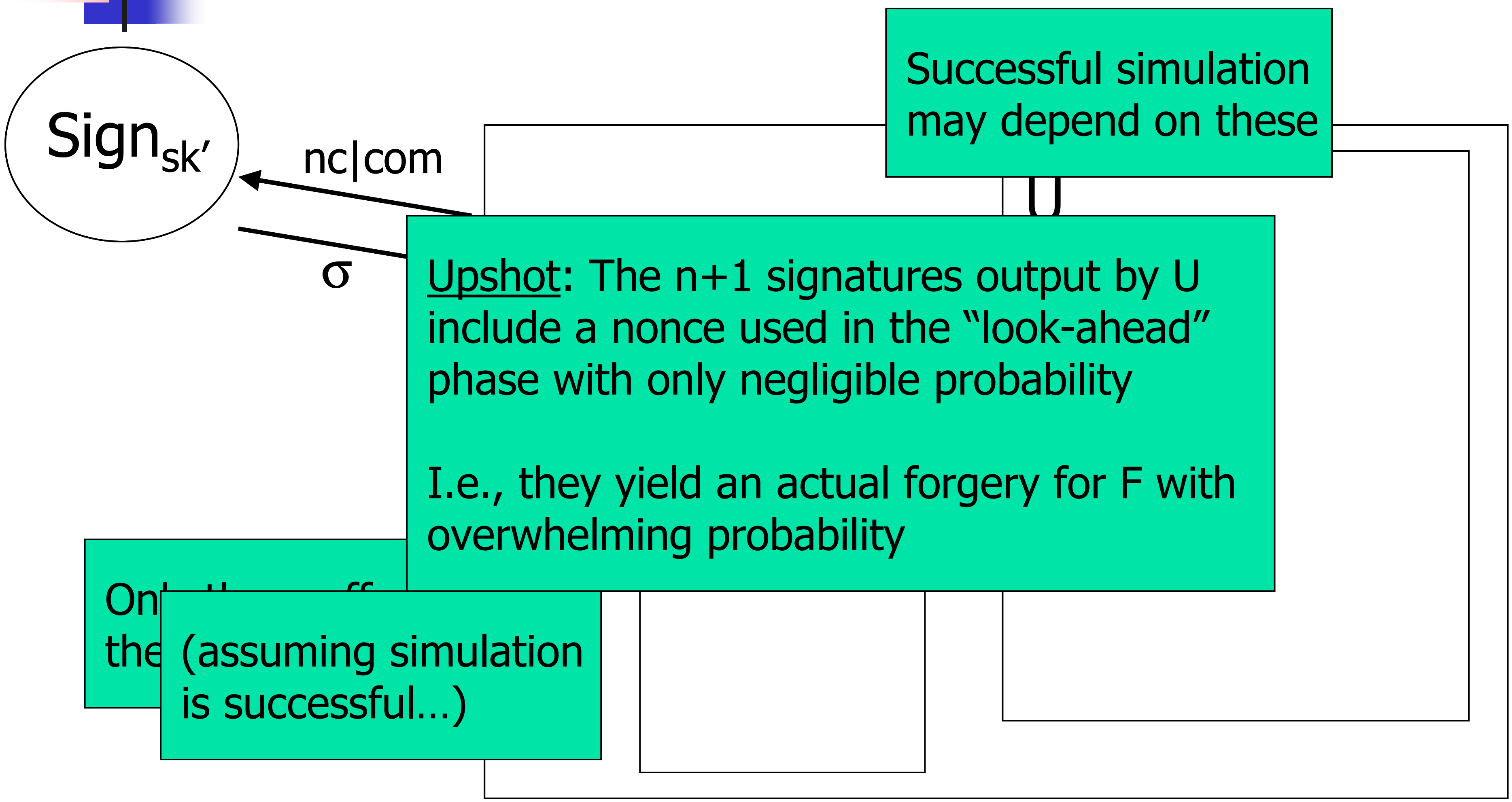
{pk* correct}

cZK: pk* correct

Analysis (unforgeability)

- Given a user who outputs n+1 forgeries:
  - Simlate cZK protocol…
  - Replace pk* by a commitment key that allows extraction…
  - With roughly equal probability, we obtain a forger F who outputs n+1 valid signatures on distinct messages
    - But makes more than n signing queries!

Analysis (unforgeability)



Analysis (blindness)

- Key point: if any sessions are successful, then pk* is (with overwhelming probability) a key that gives perfectly-hiding commitment
  - So C leaks no information!
  - Perfect hiding needed here
- By extracting the witness for pk*, can give a ZAP independent of m
- Etc…

Conclusion

- Concurrently-secure blind signatures are possible *without setup assumptions*
  - If we are satisfied with game-based definitions…
- Is the use of cZK inherent?
  - In particular, can we improve the round complexity?
  - One bottleneck is the lack of secure (standard) signatures…
- Can we hope for efficient protocols?