RingCT 2.0: A Compact Accumulator-Based (Linkable Ring Signature) Protocol for Blockchain Cryptocurrency Monero

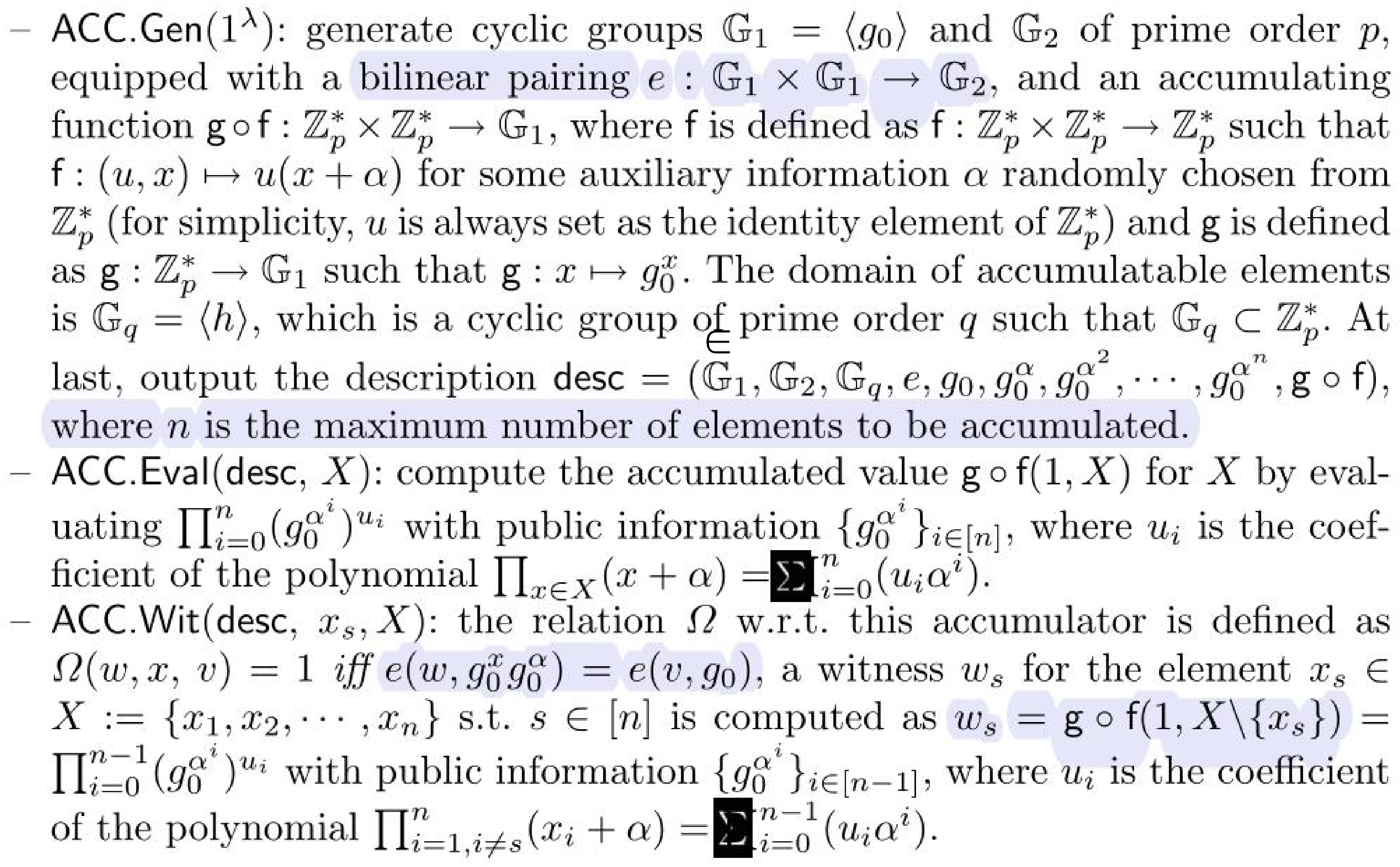
Shi-Feng Sun[1,2], Man Ho Au[1(✉)], Joseph K. Liu[3], and Tsz Hon Yuen[4]

2022-09-12

Abstract. In this work, we initially study the necessary properties and security requirements of Ring Confidential Transaction (RingCT) protocol deployed in the popular anonymous cryptocurrency Monero. Firstly, we formalize the syntax of RingCT protocol and present several formal security definitions according to its application in Monero. Based on our observations on the underlying (linkable) ring signature and commitment schemes, we then put forward a new efficient RingCT protocol (RingCT 2.0), which is built upon the well-known Pedersen commitment, accumulator with one-way domain and signature of knowledge (which altogether perform the functions of a linkable ring signature). Besides, we show that it satisfies the security requirements if the underlying building blocks are secure in the random oracle model. In comparison with the original RingCT protocol, our RingCT 2.0 protocol presents a significant space saving, namely, the transaction size is independent of the number of groups of input accounts included in the generalized ring while the original RingCT suffers a linear growth with the number of groups, which would allow each block to process more transactions.

Components - Accumulators with one-way domain

Prove x∈X has been accumulated (better than Merkle Tree)



- $\mathsf{ACC.Gen}(1^\lambda)$: generate cyclic groups $\mathbb{G}_1 = \langle g_0 \rangle$ and $\mathbb{G}_2$ of prime order p, equipped with a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, and an accumulating function $\mathsf{g} \circ \mathsf{f} : \mathbb{Z}_p^* \times \mathbb{Z}_p^* \to \mathbb{G}_1$, where f is defined as $\mathsf{f} : \mathbb{Z}_p^* \times \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ such that $\mathsf{f} : (u, x) \mapsto u(x + \alpha)$ for some auxiliary information α randomly chosen from $\mathbb{Z}_p^*$ (for simplicity, u is always set as the identity element of $\mathbb{Z}_p^*$) and g is defined as $\mathsf{g} : \mathbb{Z}_p^* \to \mathbb{G}_1$ such that $\mathsf{g} : x \mapsto g_0^x$. The domain of accumulatable elements is $\mathbb{G}_q = \langle h \rangle$, which is a cyclic group of prime order q such that $\mathbb{G}_q \subset \mathbb{Z}_p^*$. At last, output the description $\mathsf{desc} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_q, e, g_0, g_0^\alpha, g_0^{\alpha^2}, \cdots, g_0^{\alpha^n}, \mathsf{g} \circ \mathsf{f})$, where n is the maximum number of elements to be accumulated.
- $\mathsf{ACC.Eval}(\mathsf{desc}, X)$: compute the accumulated value $\mathsf{g} \circ \mathsf{f}(1, X)$ for X by evaluating $\prod_{i=0}^{n} (g_0^{\alpha^i})^{u_i}$ with public information $\{g_0^{\alpha^i}\}_{i \in [n]}$, where u_i is the coefficient of the polynomial $\prod_{x \in X}(x + \alpha) = \sum_{i=0}^{n}(u_i \alpha^i)$.
- $\mathsf{ACC.Wit}(\mathsf{desc}, x_s, X)$: the relation Ω w.r.t. this accumulator is defined as $\Omega(w, x, v) = 1$ *iff* $e(w, g_0^x g_0^\alpha) = e(v, g_0)$, a witness w_s for the element $x_s \in X := \{x_1, x_2, \cdots, x_n\}$ s.t. $s \in [n]$ is computed as $w_s = \mathsf{g} \circ \mathsf{f}(1, X \backslash \{x_s\}) = \prod_{i=0}^{n-1} (g_0^{\alpha^i})^{u_i}$ with public information $\{g_0^{\alpha^i}\}_{i \in [n-1]}$, where u_i is the coefficient of the polynomial $\prod_{i=1, i \neq s}^{n}(x_i + \alpha) = \sum_{i=0}^{n-1}(u_i \alpha^i)$.

Components - Signature of Knowledge & Pedersen Commitment

SoK: As a black box of ZKP

Pedersen Commitment:

$Com(v) = v \cdot B + \tilde{v} \cdot \tilde{B}$，其中$B, \tilde{B}$为椭圆曲线上的两个基点，$v$是需要承诺的秘密数，$\tilde{v}$为（随机）盲化因子。

具备同态加法特性，即：

$$Com(v_1) + Com(v_2) = v_1 \cdot B + \tilde{v_1} \cdot \tilde{B} + v_2 \cdot B + \tilde{v_2} \cdot \tilde{B} = (v_1 + v_2) \cdot B + (\tilde{v_1} + \tilde{v_2}) \cdot \tilde{B} = Com(v_1 + v_2)$$

ECDLP转到DLP上类似

Protocol Description

- $Setup(1^\lambda)$

  $h_0, h_1, u \in \mathbb{G}_q$

- $KeyGen$

  $(sk, pk) := (x, y = h_0^x) \in \mathbb{Z}_q \times \mathbb{G}_q$

  In the context of Monero, the public key pk is always set as a one-time address, which combining with a coin constitutes a user's account.

- $Mint(pk, a)$

  $a \in \mathbb{Z}_q$ (mint amount)

  Pedersen commitment:

  choose random $r \in \mathbb{Z}_q$, computes commitment $c = h_0^r h_1^a$

  $(cn, ck) = (c, (r, a))$

  account $act = (pk, cn)$, corresponding secret key $ask = (sk, ck)$

Protocol Description

$A = \{pk_{in,i}^{(k)}, cn_{in,i}^{(k)}\}_{i \in [n], k \in [m]}$ 👈 **Total** group of RingCT 2.0

Assume spender's group to be **s-th** group:

$A_s = \{pk_{in,s}^{(k)}, cn_{in,s}^{(k)}\}_{k \in [m]}$

- $Spend(m, K_s, A_s, A, R)$

  $K_s = \{ask_s^{(k)} = (sk_{in,s}^{(k)}, (r_{in,s}^{(k)}, a_{in,s}^{(k)}))\}$ correspond to A_s

  $R = \{pk_{out,j}\}_{j \in [t]}$ - output addresses

  - let $\sum a_{out,j} = \sum a_{in,s}^{(k)}$

    select random $r_{out,j} \in \mathbb{Z}_q$, mint new coin $cn_{out,j} = h_0^{r_{out,j}} h_1^{a_{out,j}}$

    output account $act_{out,j} = (pk_{out,j}, cn_{out,j})$

    privately send $ck_{out,j}$ to output address (use $pk_{out,j}$ to encrypt it in tx?)

  - $\widetilde{sk}_s = \sum sk_{in,s}^{(k)} + \sum r_{in,s}^{(k)} - \sum r_{out,j}$

    $\widetilde{pk}_i = \prod pk_{in,i}^{(k)} \cdot \prod cn_{in,i}^{(k)} / \prod cn_{out,j}$

    $\because \sum a_{in,s} = \sum a_{out}$

    $\therefore \widetilde{pk}_s = h_0^{\widetilde{sk}_s}$

    For convenience, let $y_i^{(m+1)} \leftarrow \widetilde{pk}_i, x_s^{(m+1)} \leftarrow \widetilde{sk}_s$

    and $y_i^{(k)} \leftarrow pk_{in,i}^{(k)}, x_s^{(k)} \leftarrow sk_{in,s}^{(k)}$

Protocol Description

- Generate proof π (SoK)

  (a) for each $k \in [m+1]$, compute

  $v_k = ACC.Eval(desc, \{y_i^{(k)} \cdot u^i\}), w_s^{(k)} = ACC.Wit(desc, \{y_i^{(k)} \cdot u^i \mid i \neq s\})$

  then we have $f(w_s^{(k)}, y_s^{(k)} \cdot u^s) = v_k$

  For convenience, let $z_s^{(k)} \leftarrow y_s^{(k)} \cdot u^s$

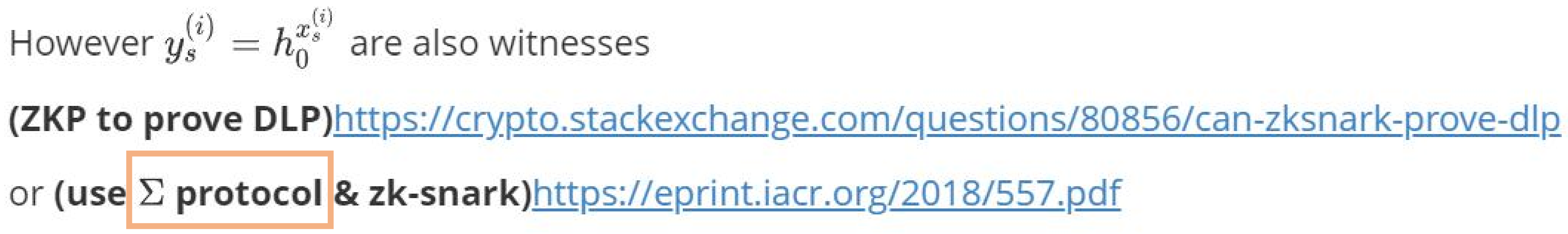
  compute sequence numbers $\underline{s_k = H(y_s^{(k)})^{x_s^{(k)}}}$

  (b) SoK pi of tx:

$$SoK\left\{\begin{array}{ll} (\{w_s^{(k)}, z_s^{(k)}, x_s^{(k)}\}_{k=1}^{m}, \gamma): & f(w_s^{(m+1)}, z_s^{(m+1)}) = v_{m+1} \wedge z_s^{(m+1)} = h_0^{x_s^{(m+1)}} u^\gamma \wedge \\ & f(w_s^{(1)}, z_s^{(1)}) = v_1 \wedge z_s^{(1)} = h_0^{x_s^{(1)}} u^\gamma \wedge s_1 = H(y_s^{(1)})^{x_s^{(1)}} \wedge \\ & \ldots \\ & f(w_s^{(m)}, z_s^{(m)}) = v_m \wedge z_s^{(m)} = h_0^{x_s^{(m)}} u^\gamma \wedge s_m = H(y_s^{(m)})^{x_s^{(m)}} \end{array}\right\}(tx)$$

Eventually, return (tx, π, S), where $S = \{s_1, s_2, \ldots, s_m\}$. We note that the serial number s_k is uniquely determined by the address key $sk_{in,s}^{(k)}$ for every $k \in [m]$, and thus they can be used to prevent double-spending.

TODO



However $y_s^{(i)} = h_0^{x_s^{(i)}}$ are also witnesses

(ZKP to prove DLP)https://crypto.stackexchange.com/questions/80856/can-zksnark-prove-dlp

or **(use $\boxed{\Sigma \text{ protocol}}$ & zk-snark)**https://eprint.iacr.org/2018/557.pdf

[效率问题] - zcash完全基于zk-snark进行隐私保护，而RingCT 2.0基于环签名（同样有ZKP环节）来使得sender无法追踪receiver接下来的资金流向，效率未对比

[内存问题] - Monero根据S的唯一性来防止双花，但RingCT 2.0使得无法区分哪些act(account)已被消耗，即无法类似Bitcoin可删除已消耗的UTXO

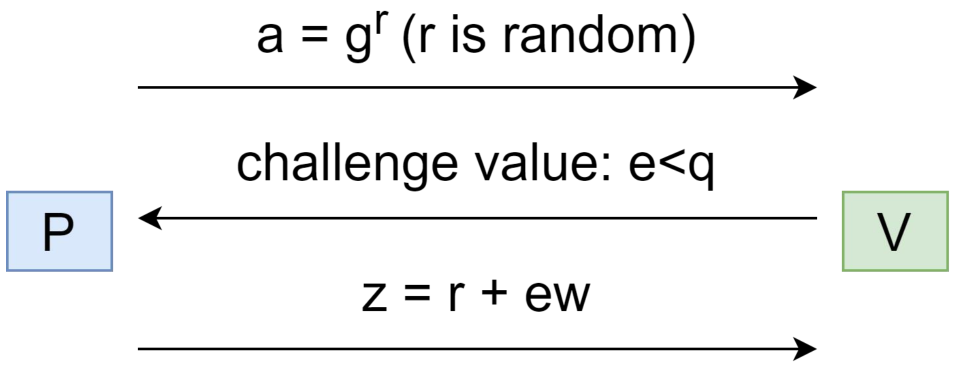
Σ-protocol

Given an example based on Schnorr:

p, q is prime, $q|(p-1)$

$g \in Z_p^*, ord(g) = q$

Prover has chosen the witness $w \in Z_q$, and has published $h = g^w \ mod\ p$



(1) g^z ?= ah^e
(2) ord(h) ?= ord(g) ?= q

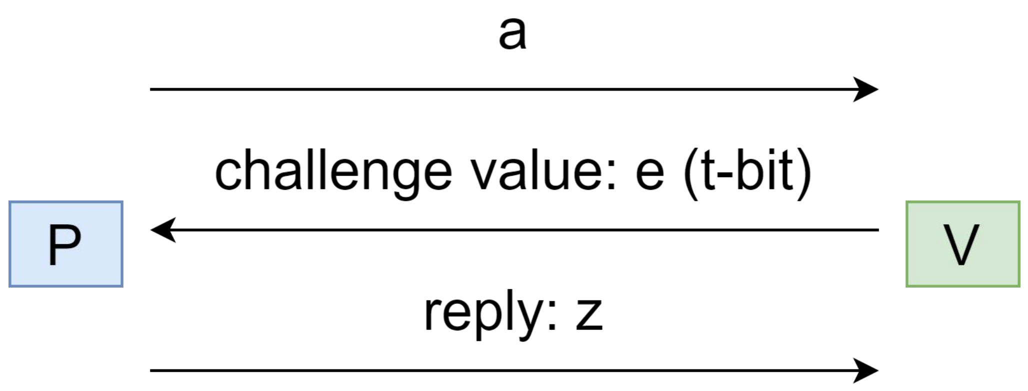
(Concretely, e\in Z_{2^t}, 2^t < q, i.e. size(e) <= t)

Σ-protocol

General properties for Σ-protocol:

* R: binary relation (subset of {0, 1}*×{0, 1}*), (x, w)∈R
* x: instance of some computational problem （e.g. 前文基于Schnorr的Σ协议即为DLP）
* w: solution of x (witness)

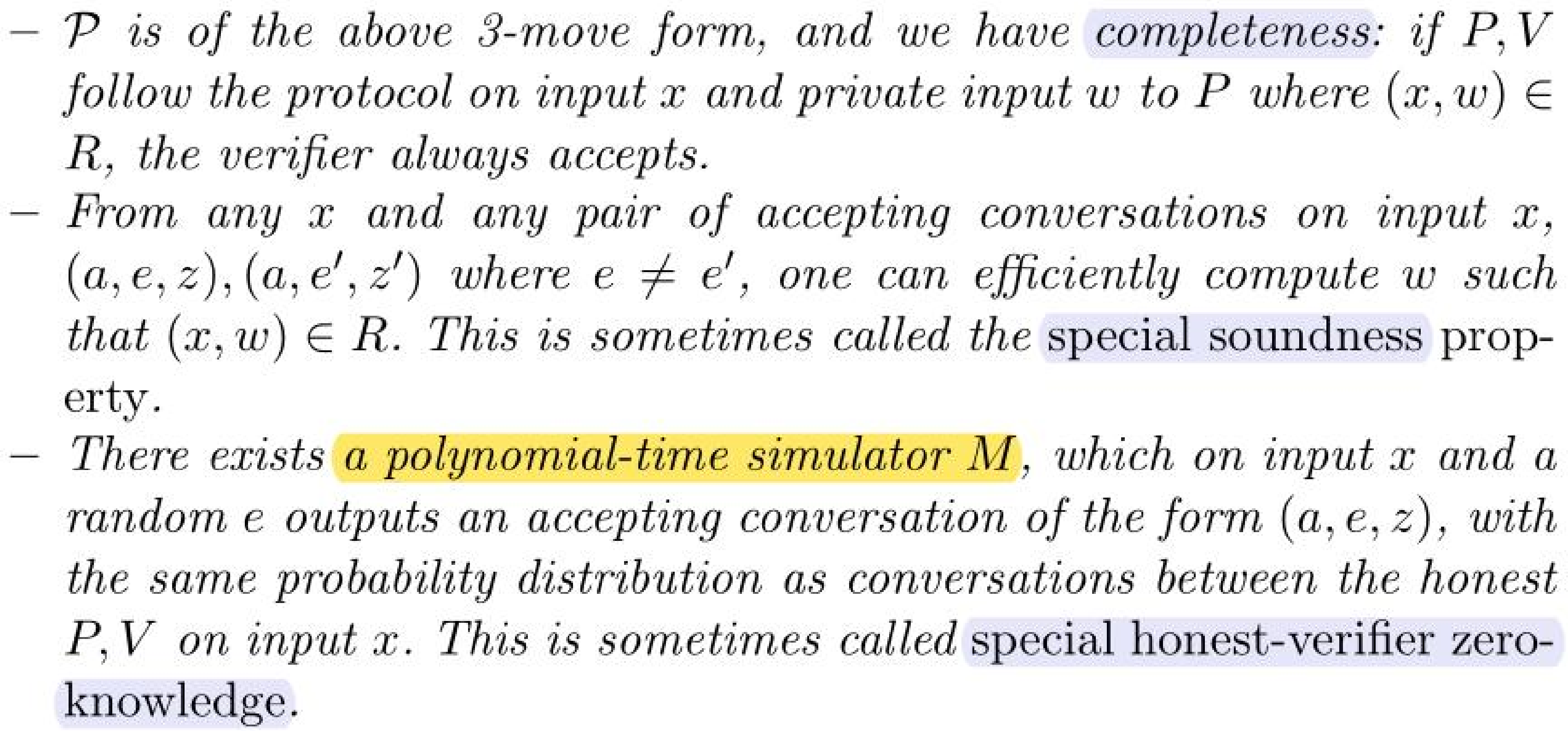
一般性sigma协议图示:



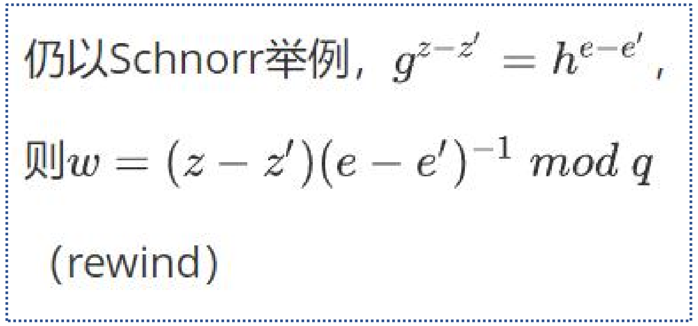
根据元组(x, a, e, z)决定是否接受P (Prover)

Σ-protocol

不同于一般的zkp，Σ协议满足下列形式：



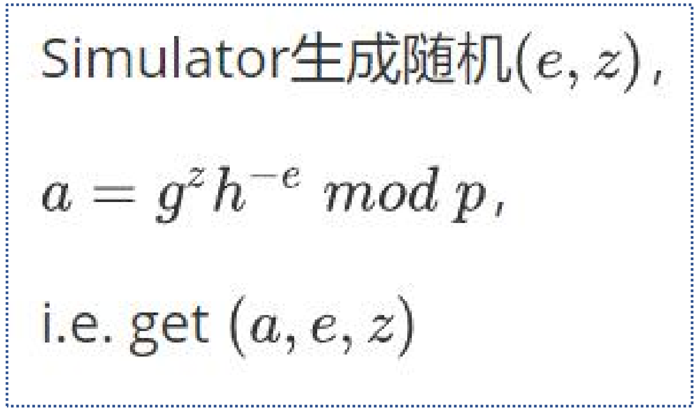
- $\mathcal{P}$ *is of the above 3-move form, and we have* completeness*: if P, V follow the protocol on input x and private input w to P where $(x, w) \in R$, the verifier always accepts.*
- *From any x and any pair of accepting conversations on input x, $(a, e, z), (a, e', z')$ where $e \neq e'$, one can efficiently compute w such that $(x, w) \in R$. This is sometimes called the* special soundness property.
- *There exists a polynomial-time simulator M, which on input x and a random e outputs an accepting conversation of the form (a, e, z), with the same probability distribution as conversations between the honest P, V on input x. This is sometimes called* special honest-verifier zero-knowledge.



仍以Schnorr举例，$g^{z-z'} = h^{e-e'}$，

则$w = (z - z')(e - e')^{-1} \bmod q$

（rewind）



Simulator生成随机(e, z)，

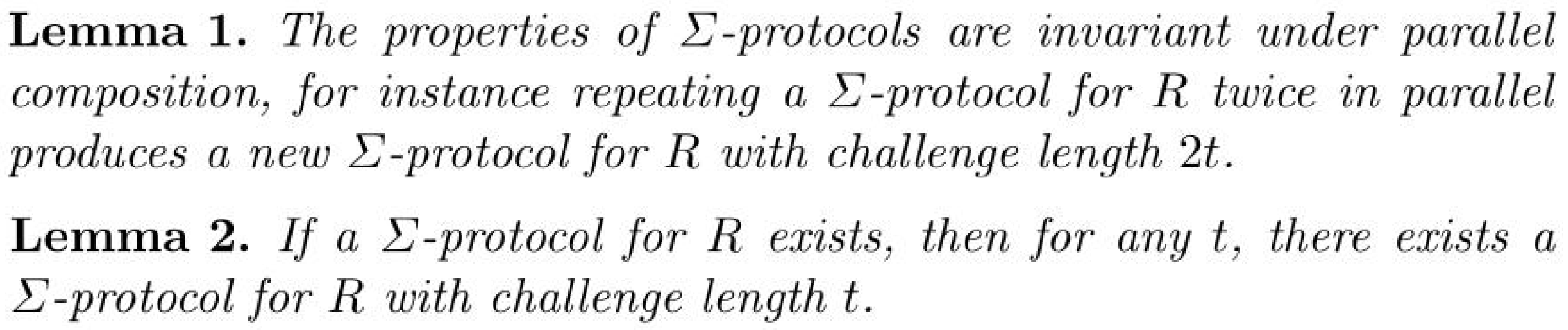
$a = g^{z} h^{-e} \bmod p$，

i.e. get (a, e, z)

[!] 模拟器分布相同
[!] 误差概率为2^{-t}

Σ-protocol

Σ-protocol can be parallelized:


Lemma 1. *The properties of Σ-protocols are invariant under parallel composition, for instance repeating a Σ-protocol for R twice in parallel produces a new Σ-protocol for R with challenge length $2t$.*

Lemma 2. *If a Σ-protocol for R exists, then for any t, there exists a Σ-protocol for R with challenge length t.*

t\nmid size(t’)时，在并行的某个挑战中将e padding若干位0即可

[!] 挑战长度过大（过多并行）时，Σ-protocol将失去零知识性（增大无法rewind的概率）
e.g. ∃ q | (e - e’), 逆元不存在