Introduction to Cryptography
Lecture 12

Secret sharing
Electronic cash

Benny Pinkas

Secret Sharing

- 3-out-of-3 secret sharing:
  - Three parties, A, B and C.
  - Secret *S.*
  - No two parties should know anything about *K*, but all three together should be able to retrieve it.
- In other words
  - A + B + C $\Rightarrow$ S
  - But,
    - A + B $\not\Rightarrow$ S
    - A + C $\not\Rightarrow$ S
    - B + C $\not\Rightarrow$ S

Secret Sharing

- 3-out-of-3 secret sharing:
- How about the following scheme:
  - Let $S=s_1s_2\ldots s_m$ be the bit representation of S. (m is a multiple of 3)
    - Party A receives $s_1,\ldots,s_{m/3}$.
    - Party B receives $s_{m/3+1},\ldots,s_{2m/3}$.
    - Party C receives $s_{2m/3+1},\ldots,s_m$.
  - All three parties can recover S.
  - Why doesn't this scheme satisfy the definition of secret sharing?

Secret Sharing

- Solution:
  - Define *shares* for A,B,C in the following way
  - (S_A, S_B, S_C) is a random triple, subject to the constraint that
    - $S_A \oplus S_B \oplus S_C = S$
    - *or,* S_A and S_B are random, and $S_C = S_A \oplus S_B \oplus S$.

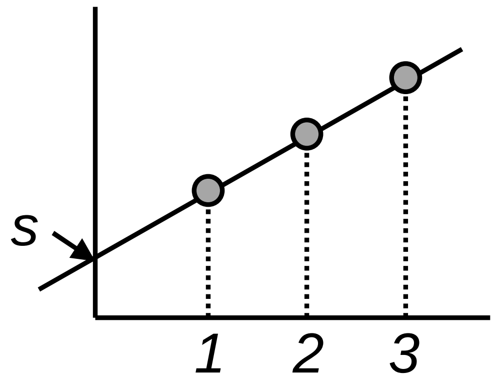
- What if it is required that any one of the parties should be able to compute S?
  - Set $S_A = S_B = S_C = S$

t-out-of-*n* secret sharing

- Provide shares to *n* parties, satisfying
  - Recoverability: any *t* shares enable the reconstruction of the secret.
  - Secrecy: any *t-1* shares reveal nothing about the secret.

- We saw *1*-out-of-*n* and *n*-out-of-*n* secret sharing.

- Consider *2*-out-of-*n* secret sharing.
  - Define a line which intersects the Y axis at *S*
  - The shares are points on the line
  - Any two shares define *S*
  - A single share reveals nothing



t-out-of-*n* secret sharing

- Fact: Let F be a field. Any $d+1$ pairs (a_i, b_i) define a unique polynomial P of degree $\leq d$, s.t. $P(a_i)=b_i$. (assuming $d < |F|$).

- Shamir's secret sharing scheme:
  - Choose a large prime and work in the field Zp.
  - The secret S is an element in the field.
  - Define a polynomial P of degree $t-1$ by choosing random coefficients $a_1,\ldots,a_{t-1}$ and defining
    $P(x) = a_{t-1}x^{t-1}+\ldots+a_1x+\underline{S}$.
  - The share of party j is $(j, P(j))$.

t-out-of-n secret sharing

- Reconstruction of the secret:
  - Assume we have $P(x_1),\ldots,P(x_t)$.
  - Use Lagrange interpolation to compute the unique polynomial of degree ≤ $t-1$ which agrees with these points.
  - Output the free coefficient of this polynomial.

- Lagrange interpolation
  - $P(x) = \sum_{i=1..t} P(x_i)\cdot L_i(x)$
  - where $L_i(x)=\prod_{j\neq i}(x-x_j) / \prod_{j\neq i}(x_i-x_j)$
  - (Note that $L_i(x_i)=1$, $L_i(x_j)=0$ for $j\neq i$.)

  - I.e., $S = \sum_{i=1..t} P(x_i) \cdot \prod_{j\neq I} x_j \;/\; \prod_{j\neq i}(x_i - x_j)$

Properties of Shamir's secret sharing

- Perfect secrecy: Any *t-1* shares give no information about the secret: $\Pr(secret=s \mid P(1),\ldots,P(t-1)) = \Pr(secret=s)$. (Security is not based on any assumptions.)
- Proof:
  - The polynomial is generated by choosing a random polynomial of degree *t-1,* subject to *P(0)*=secret.
  - Suppose that the shares are $P(x_1),\ldots,P(x_{t-1})$.
  - *P()* is generated by choosing uniformly random values to the *t-1* coefficients, $a_1,\ldots,a_{t-1}$. (a_0 is already set to be *S)*
    - Any assignment of values to $a_1,\ldots,a_{t-1}$ defines a single set of values to $P(x_1),\ldots,P(x_{t-1})$.
    - Therefore the values of $P(x_1),\ldots,P(x_{t-1})$ are uniformly distributed.

Additional properties of Shamir's secret sharing

- Ideal size: Each share is the same size as the secret.
- Extendable: Additional shares can be easily added.
- Flexible: different weights can be given to different parties by giving them more shares.
- Homomorphic property: Suppose $P(1),\dots,P(n)$ are shares for S, and $P'(1),\dots,P'(n)$ are shares for S', then $P(1)+P'(1),\dots,P(n)+P'(n)$ are shares for $S+S'$.

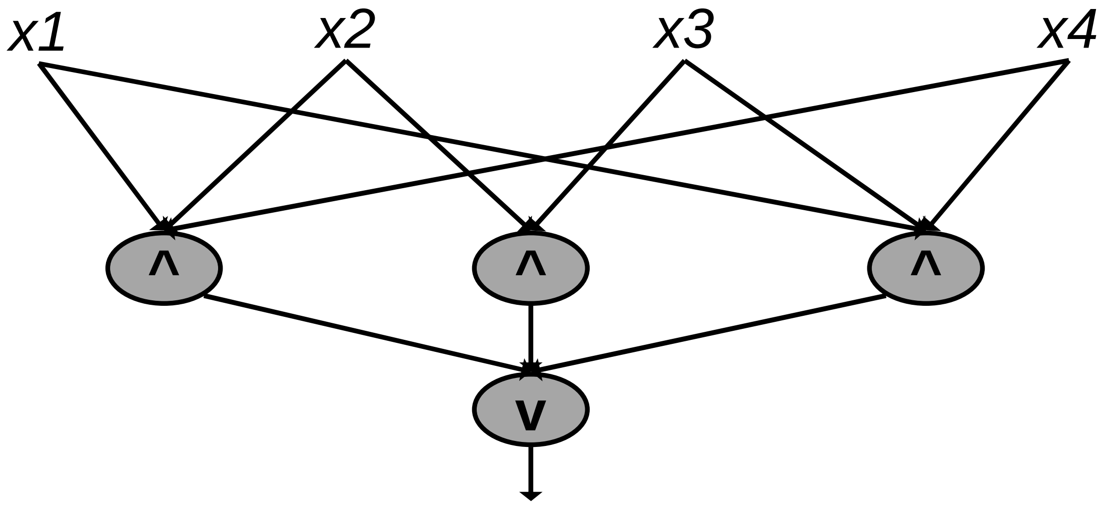
General secret sharing

- P is the set of users (say, n users).
- $A \in \{1,2,\ldots,n\}$ is an authorized subset if it is authorized to access the secret.
- Γ is the set of authorized subsets.
- For example,
  - $P = \{1,2,3,4\}$
  - Γ = *Any set containing one of* $\{ \{1,2,4\}, \{1,3,4,\}, \{2,3\} \}$
  - Not supported by threshold secret sharing

- If $A \in \Gamma$ and $A \subseteq B$, then $B \in \Gamma$.
- $A \in \Gamma$ is a minimal authorized set if there is no $C \subseteq A$ such that $C \in \Gamma$.
- The set of minimal subsets Γ_0 is called the basis of Γ.

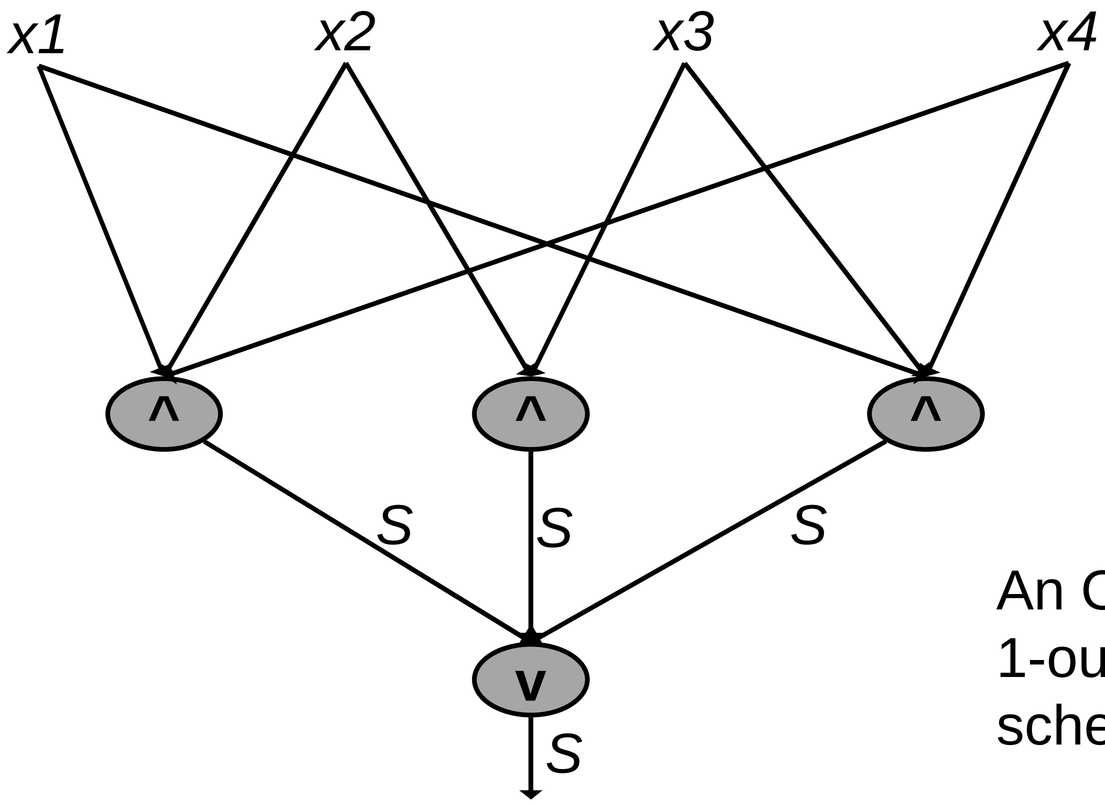
The monotone circuit construction (Benaloh-Leichter)

- A Boolean circuit C with OR and AND gates, is *monotone.* Namely, if *C(x)=1*, then changing bits of *x* from 0 to 1 does not change the result to 0.
- Given Γ construct a circuit C s.t. *C(A)=1* iff $A \in \Gamma$.
  - $\Gamma_0 = \{ \{1,2,4\}, \{1,3,4,\}, \{2,3\} \}$



Handling OR gates

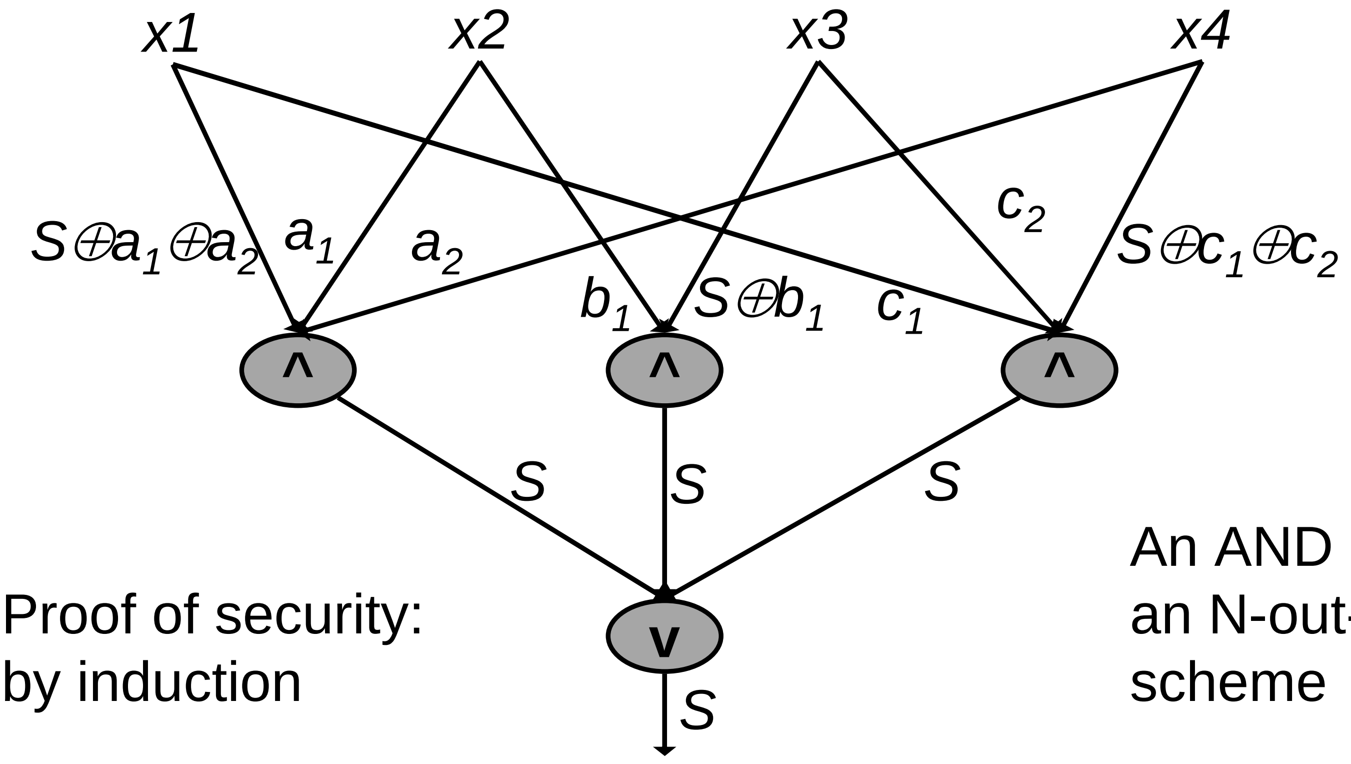
Starting from the output gate and going backwards



An OR gate is a 1-out-of-N scheme

Handling AND gates

Final step: each user gets the keys of the wires going out from its variable

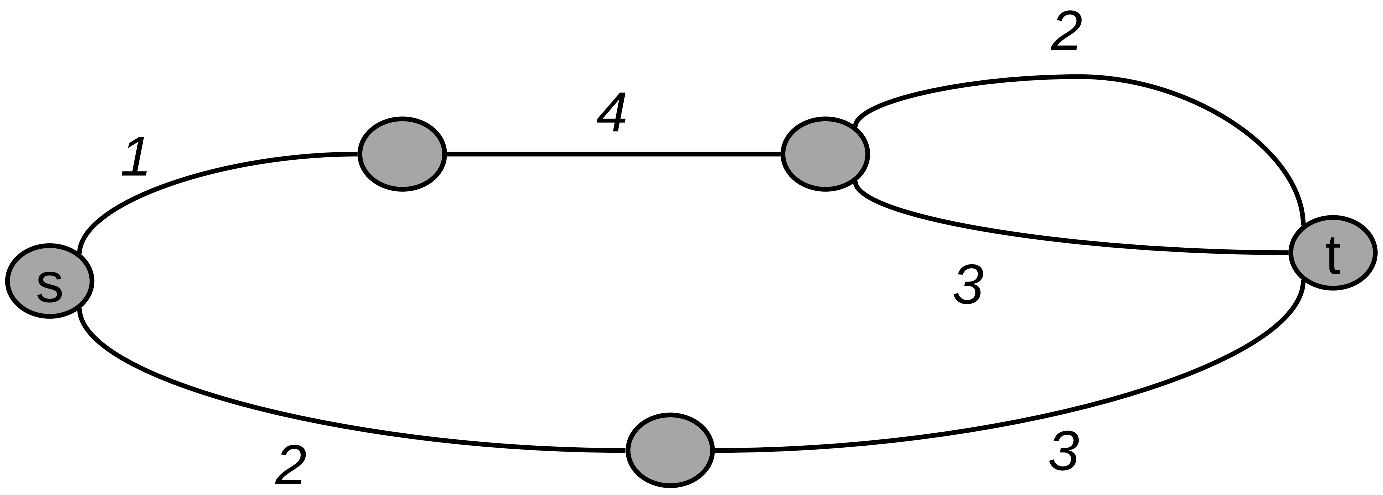


Proof of security: by induction

An AND gate is an N-out-of-N scheme

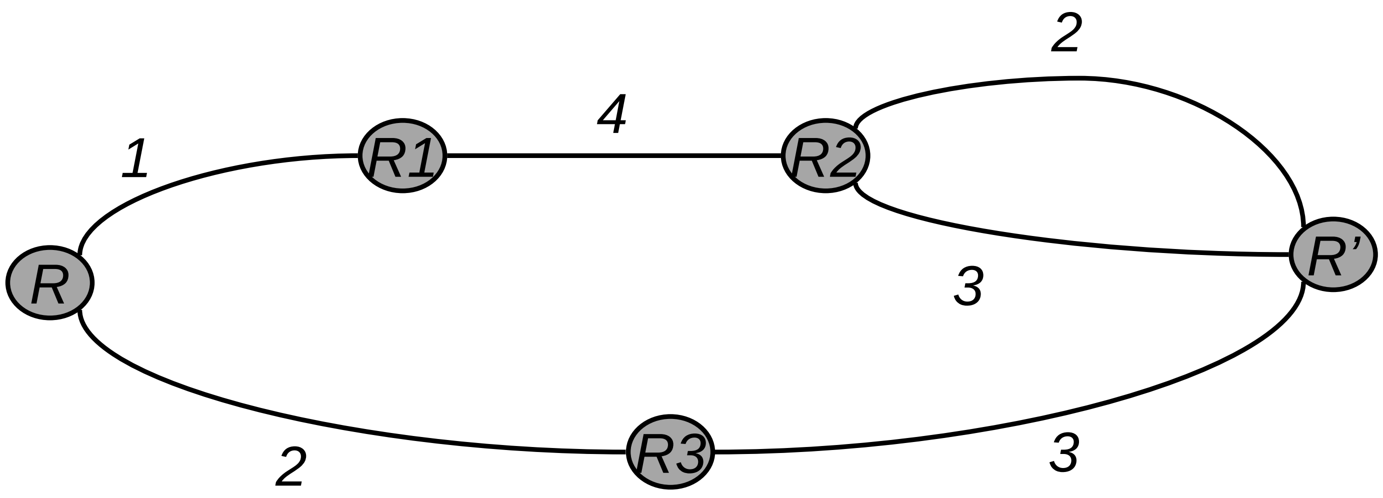
A graph based construction

- Represent the access structure by an undirected graph.
- An authorized set corresponds to a path from s to t in an undirected graph.
- $\Gamma_0 = \{ \{1,2,4\}, \{1,3,4,\}, \{2,3\} \}$

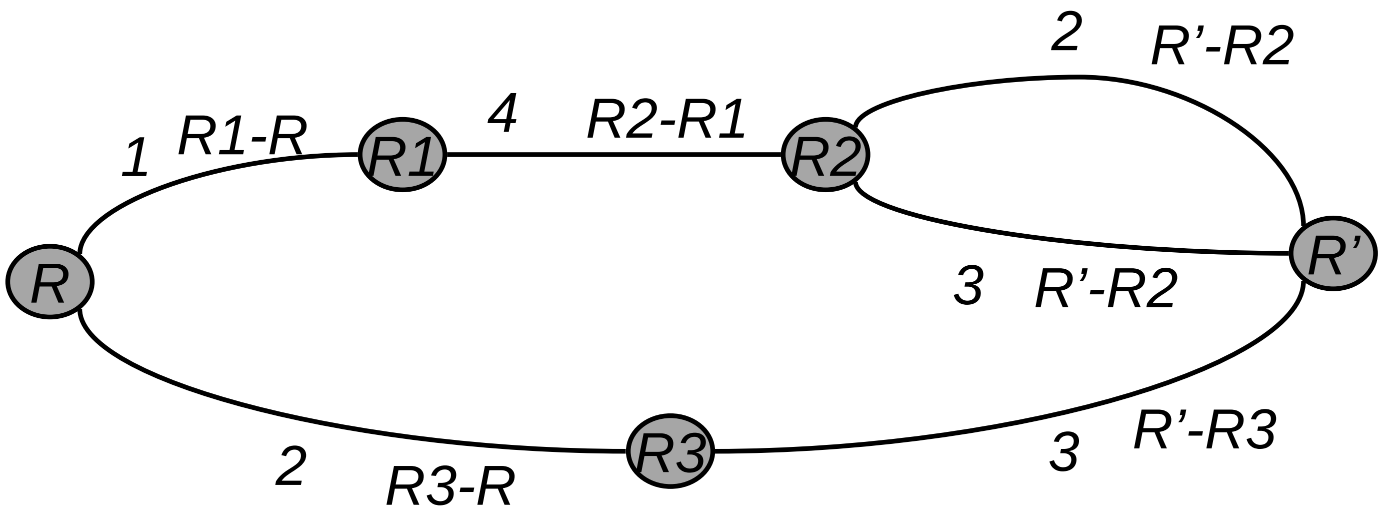


A graph based construction

Assign random values to nodes, s.t. *R'*-*R*= *shared secret* *(R'=R+shared secret)*



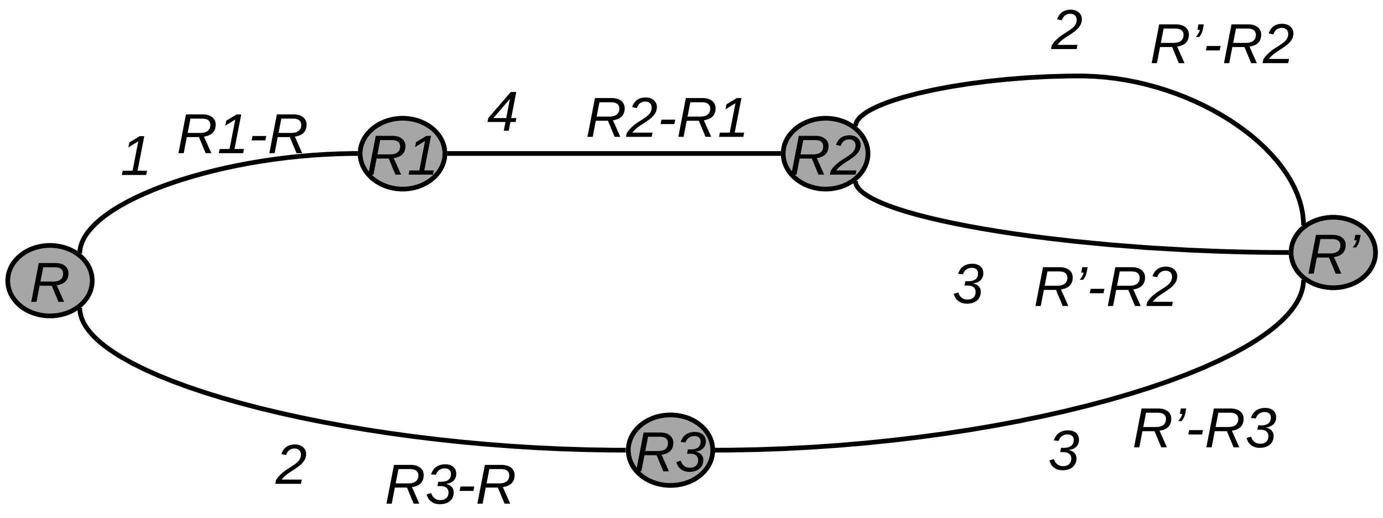
A graph based construction



- Assign to edge R1→R2 the value *R2-R1*
- Give to each user the values associated with its edges

A graph based construction

- Consider the set {1,2,4}
- why can an authorized set reconstruct the secret? Why can't a unauthorized set do that?



Electronic cash

Simple electronic checks

- A payment protocol:
  - Sign a document transferring money from your account to another account
  - This document goes to your bank
  - The bank verifies that this is not a copy of a previous check
  - The bank checks your balance
  - The bank transfers the sum

- Problems:
  - Requires online access to the bank (to prevent reusage)
  - Expensive.
  - The transaction is traceable (namely, the bank knows about the transaction between you and Alice).

First try at a payment protocol

- Withdrawal
  - User gets bank signature on {I am a $100 bill, #1234}
  - Bank deducts $100 from user's account
- Payment
  - User gives the signature to a merchant
  - Merchant verifies the signature, and checks online with the bank to verify that this is the first time that it is used.
- Problems:
  - As before, online access to the bank, and lack of anonymity.
- Advantage:
  - The bank doesn't have to check online whether there is money in the user's account.
  - In fact, there is no real need for the signature, since the bank checks its own siganture.

Anonymous cash via blind signatures

- The bank signs the bill without seeing it
  - (e.g. like signing on a carbon paper)

- RSA Blind signatures (Chaum)
- RSA signature: $(H(m))^{1/e}$ mod n
- Blind RSA signature:
  - Alice sends Bob ($r^e H(m)$) mod n, where r is a random value.
  - Bob computes $(r^e H(m))^{1/e} = r\, H(m)^{1/e}$ mod n, and sends to Alice.
  - Alice divides by r and computes $H(m)^{1/e}$ mod n

- Problem: Alice can get Bob to sign anything, Bob does not know what he is signing.

Enabling the bank to verify the signed value

- "cut and choose" protocol
- Suppose Alice wants to sign a $20 bill.
  - A $20 bill is defined as *H(random index,$20)*.
  - Alice sends to bank 100 different $20 bills for blind signature.
  - The bank chooses 99 of these and asks Alice to unblind them (divide by the corresponding *r* values). It verifies that they are all $20 bills.
  - The bank blindly signs the remaining bill and gives it to Alice.
  - Alice can use the bill without being identified by the bank.

- If Alice tries to cheat she is caught with probability 99/100.
- 100 can be replaced by any parameter *m*.
- We would like to have an exponentially small cheating probability.

Exponentially small cheating probability

- Define that a \$20 bill is valid if it is the e^{th} root of the multiplication of 50 values of the form *H(x),* (where *x="random index,\$20"*), and the owner of the bill can present all 50 *x* values.

- The withdrawal protocol:
  - Alice sends to the Bank $z_1, z_2, \dots, z_{100}$ (where $z_i = r_i^{\,e} \cdot H(x_i)$).
  - The Bank asks Alice to reveal ½ of the values $z_i = r_i^{\,e} \cdot H(x_i)$.
  - The Bank verifies them and extracts the e^{th} root of the multiplication of all the other 50 values.
- Payment: Alice sends the signed bill and reveals the 50 preimage values. The merchant sends them to the bank which verifies that they haven't been used before.

Online vs. offline digital cash

- We solved the anonymity problem, while verifying that Alice can only get signatures on bills of the right value.

- The bills can still be duplicated
- Merchants must check with the bank whenever they get a new bill, to verify that it wasn't used before.

- A new idea:
  - During the payment protocol the user is forced to encode a random identity string (RIS) into the bill
  - If the bill is used twice, the RIS reveals the user's identity and she loses her anonymity.

Offline digital cash

Withdrawal protocol:

- Alice prepares 100 bills of the form
  - {I am a \$20 bill, *#1234*, $y_1,y'_1,y_2,y'_2,\ldots,y_m,y'_m$}
  - S.t. $\forall\, i\ y_i=H(x_i)$, $y'_i=H(x'_i)$, and it holds that $x_i \oplus x'_i$ =*Alice's id,* where *H()* is a collision resistant function.
- Alice blinds these bills and sends to the bank.
- The bank asks her to unblind 99 bills and show their x_i,x'_i values, and checks their validity. (Alternatively, as in the previous example, Alice can do a check with fails with only an exponential probability.)
- The bank signs the remaining blinded bill.

Offline digital cash

Payment protocol:

- Alice gives a signed bill to the vendor
  - {I am a \$20 bill, #*1234*, $y_1, y'_1, y_2, y'_2, \ldots, y_m, y'_m$}
- The vendor verifies the signature, and if it is valid sends to Alice a random bit string b=$b_1b_2\ldots b_m$ of length m.
- $\forall i$ if $b_i=0$ *Alice returns* x_i, otherwise ($b_i=1$) she returns x'_I
- The vendor checks that $y_i=H(x_i)$ or $y'_i=H(x'_i)$ (depending on b_i). If this check is successful it accepts the bill. (Note that Alice's identity is kept secret.)

- Note that the merchant does not need to contact the bank during the payment protocol.

Offline digital cash

- The merchant must deposit the bill in the bank. It cannot use the bill to pay someone else.
  - Because it cannot answer challenges *b** different from the challenge *b* it sent to Alice.

- How can the bank detect double spenders?
  - Suppose two merchants *M* and M^* receive the same bill
  - With very high probability, they send *different* queries *b,b**
  - There is an index *i* for which $b_i=0$, $b^*_i=1$. Therefore *M* receives x_i and *M** receives x'_i.
  - When they deposit the bills the bank receives x_i and x^*_i, and can compute $x_i \oplus x'_i$ *=Alice's id.*