Introduction to Cryptography
Lecture 12

Benny Pinkas

- Some practical issues in number theory

- Last week
  - Primality testing
  - Pollard's rho method for factoring

Integer factorization

- The RSA and Rabin cryptosystems use a modulus N and are insecure if it is possible to factor N.

- Factorization: given N find all prime factors of N.

- Factoring is the search problem corresponding to the primality testing decision problem.
  - Primality testing is easy
  - What about factoring?

Pollard's Rho method

- Factoring *N*

- Trivial algorithm: trial division by all integers < $N^{1/2}$.

- Pollard's rho method:
  - $O(N^{1/4})$ computation.
  - O(1) memory.
  - A heuristic algorithm.

Modern factoring algorithms

- The number-theoretic running time function $L_n(a,c)$

$$L_n(a,c) = e^{c(\ln n)^a (\ln\ln n)^{1-a}}$$

  - For a=0, the running time is polynomial in ln(n).
  - For a=1, the running time is exponential in ln(n).
  - For 0<a<1, the running time is subexponential.

- Factoring algorithms
  - Quadratic field sieve: $L_n(1/2, 1)$
  - General number field sieve: $L_n(1/3, 1.9323)$
  - Elliptic curve method $L_p(1/2, 1.41)$ (preferable only if p<<sqrt(n))

Modulus size recommendations

- Factoring algorithms are run on massively distributed networks of computers (running in their idle time).
- RSA published a list of factoring challenges.
- A 512 bit challenge was factored in 1999.
- The largest factored number $n=pq$.
  - 768 bits (RSA-768)
  - Factored on January 7, 2010 using the NFS

- Typical current choices:
  - At least 1024-bit RSA moduli should be used
  - For better security, longer RSA moduli are used
  - For more sensitive applications, key lengths of 2048 bits (or higher) are used

RSA with a modulus with more factors

- The best factoring algorithms:
  - General number field sieve (NFS): $L_n(1/3, 1.9323)$
  - Elliptic curve method $L_p(1/2, 1.41)$

- If n=pq, where |p|=|q|, then the NFS is faster.
  - This is true even though $p=n^{1/2}$.
  - Common parameters: |p|=|q|=512 bits
  - Factoring using the NFS is infeasible, but more likely than factoring using the elliptic curve method.

RSA for paranoids

- Suppose $N=pq$, $|p|=500$ bits, $|q|=4500$ bits.
- Factoring is extremely hard.
  - The NFS has to be applied to a much larger modulus. The elliptic curve method is still inefficient.
- Decryption is also very slow. (Encryption is done using a short exponent, so it is pretty efficient.)

- However, in most applications RSA is used to transfer session keys, which are rather short.
- Assume message length is < 500 bits.
  - In the decryption process, it is only required to decrypt the message modulo p. (As, or more, efficient, as a 1024 bit n.)
  - Encryption must use a slightly longer *e*. Say, e=20.

Discrete log algorithms

- Input: *(g,y)* in a finite group G. Output: x s.t. $g^x = y$ in G.
- Generic vs. special purpose algorithms: generic algorithms do not exploit the representation of group elements.

- Algorithms
  - Baby-step giant-step: Generic. |G| can be unknown. Sqrt(|G|) running time and memory.
  - Pollard's rho method: Generic. |G| must be known. Sqrt(|G|) running time and O(1) memory.
  - No generic algorithm can do better than O(sqrt(q)), where q is the largest prime factor of |G|
  - Pohlig-Hellman: Generic. |G| and its factorization must be known. O(sqrt(q) ln q), where q is largest prime factor of |G|.
  - Therefore for Z^*_p, p-1 must have a large prime factor.
  - Index calculus algorithm for Z^*_p: L(1/2, c)
  - Number field size for Z^*_p: L(1/3, 1.923)

Elliptic Curves

- The best discrete log algorithm which works even if |G| can be unknown is the baby-step giant-step algorithm.
  - Sqrt(|G|) running time and memory.
- Other (more efficient) algorithms must know |G|.
  - In Z_p* we know that | Z_p* |=p-1.

- Elliptic curves are groups G where
  - The Diffie-Hellman assumption is assumed to hold, and therefore we can run DH an ElGamal encryption/sigs.
  - |G| is unknown and therefore the best discrete log algorithm us pretty slow
  - It is therefore believed that a small Elliptic Curve group is as secure as larger Z_p* group.
  - Smaller group -> smaller keys and more efficient operations.

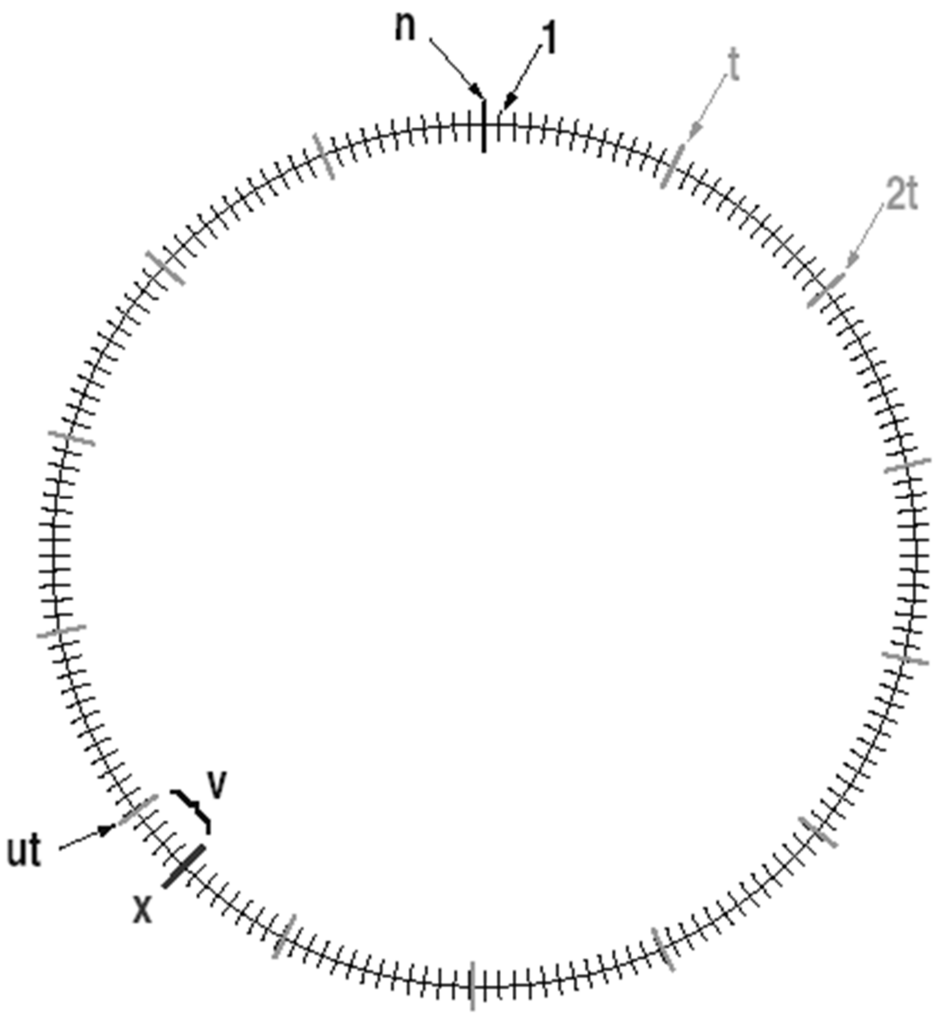
Baby-step giant-step DL algorithm

- Let t=sqrt(|G|).
- x can be represented as $x=ut-v$, where u,v < sqrt(|G|).

- The algorithm:
  - Giant step: compute the pairs $(j, g^{j\cdot t})$, for $0 \le j \le t$. Store in a table keyed by $g^{j\cdot t}$.
  - Baby step: compute $y\cdot g^i$ for $i=0,1,2\ldots$, until you hit an item $(j, g^{j\cdot t})$ in the table. x = $jt - i$.

- Memory and running time are O(sqrt|G|).

Baby-step giant-step DL algorithm



Secret sharing

Secret Sharing

- 3-out-of-3 secret sharing:
  - Three parties, A, B and C.
  - Secret *S.*
  - No two parties should know *anything* about *S*, 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 $S=s_1 s_2 \ldots s_m$ 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?
  - Why does each share need to be as long as the secret?

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 $S_A = S_B = S_C = S$

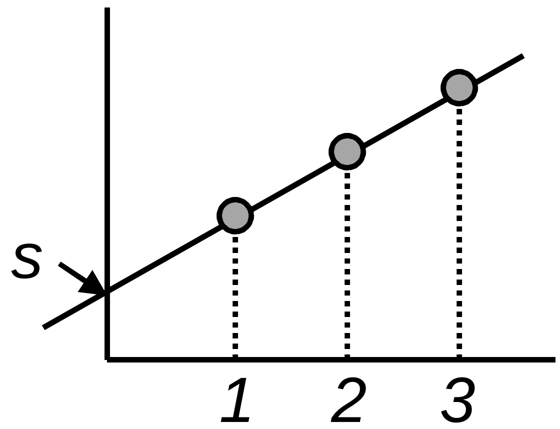
- What if each pair of the three parties should be able to compute S?

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 ≤ 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 $t\text{-}1$ 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:
  - Let's get intuition from 2-out-of-n secret sharing
  - The polynomial is generated by choosing a random coefficient *a* and defining $P(x)= a\cdot x+s$.
  - Suppose that the adversary knows $P(x_1)=a\cdot x_1+s$.
  - For any value of *s*, the value of *a* is uniquely defined by $P(x_1)$ and *s*.
  - Namely, $\forall s$ there is one-to-one correspondence between *a* and $P(x_1)$.
  - Since *a* is uniformly distributed, so is the value of $P(x_1)$ (any assignment to *a* results in exactly one value of $P(x_1)$).
    - Therefore $P(x_1)$ does not reveal any information about *s*.

Properties of Shamir's secret sharing

- Perfect secrecy: Any t-1 shares give no information about the secret: $\Pr(secret=s \mid P(1),\ldots,P(t\text{-}1)) = \Pr(secret=s)$.
  (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 adversary knows the shares $P(x_1),\ldots,P(x_{t-1})$.
  – The values of $P(x_1),\ldots,P(x_{t-1})$ are defined by t-1 linear equations of $a_1,\ldots,a_{t-1}$, s.
    - $P(x_i) = \Sigma_{i=1,\ldots,t-1} (x_i)^j a_j + s$.

Properties of Shamir's secret sharing

- Proof (cont.):
  - The values of $P(x_1),\ldots,P(x_{t-1})$ are defined by $t\text{-}1$ linear equations of $a_1,\ldots,a_{t-1}, s$.
    - $P(x_i) = \Sigma_{j=1,\ldots,t-1} (x_i)^j a_j + s.$
  - For any possible value of s, there is a exactly one set of values of $a_1,\ldots,a_{t-1}$ which gives the values $P(x_1),\ldots,P(x_{t-1})$.
    - This set of $a_1,\ldots,a_{t-1}$ can be found by solving a linear system of equations.
  - Since $a_1,\ldots,a_{t-1}$ *are* uniformly distributed, so are the values of $P(x_1),\ldots,P(x_{t-1})$.
    - Therefore $P(x_1),\ldots,P(x_{t-1})$ reveal nothing about s.

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 $P(1),\ldots,P(n)$ are shares of S, and $P'(1),\ldots,P'(n)$ are shares of S', then $P(1)+P'(1),\ldots,P(n)+P'(n)$ 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.

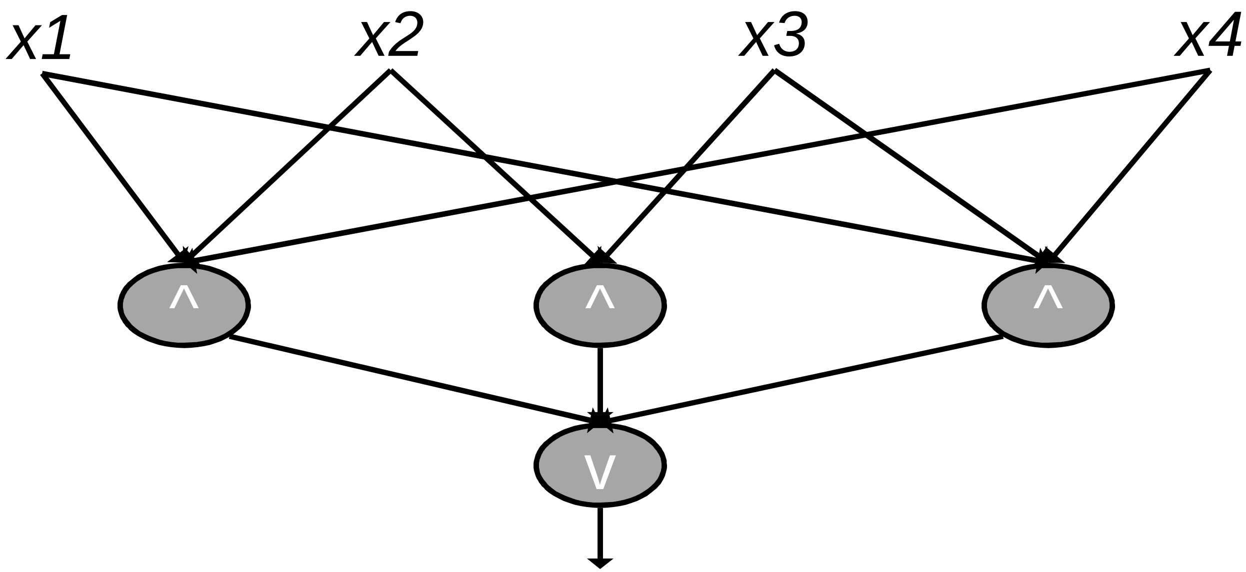
Why should we examine general access structures?

- Some general access structures can be implemented using threshold access structures.
- But not all access structures can be represented by threshold access structures

- For example, consider the access structure $\Gamma=\{\{1,2\},\{3,4\}\}$
  - Any threshold based secret sharing scheme with threshold t gives weights to parties, such that $w_1+w_2 \geq t$, and $w_3+w_4 \geq t$.
  - Therefore either $w_1 \geq t/2$, or $w_2 \geq t/2$. Suppose that this is w_1.
  - Similarly either $w_3 \geq t/2$, or $w_4 \geq t/2$. Suppose that this is w_3.
  - In this case parties 1 and 3 can reveal the secret, since $w_1+w_3 \geq t$.
  - Therefore, this access structure cannot be realized by a threshold scheme.

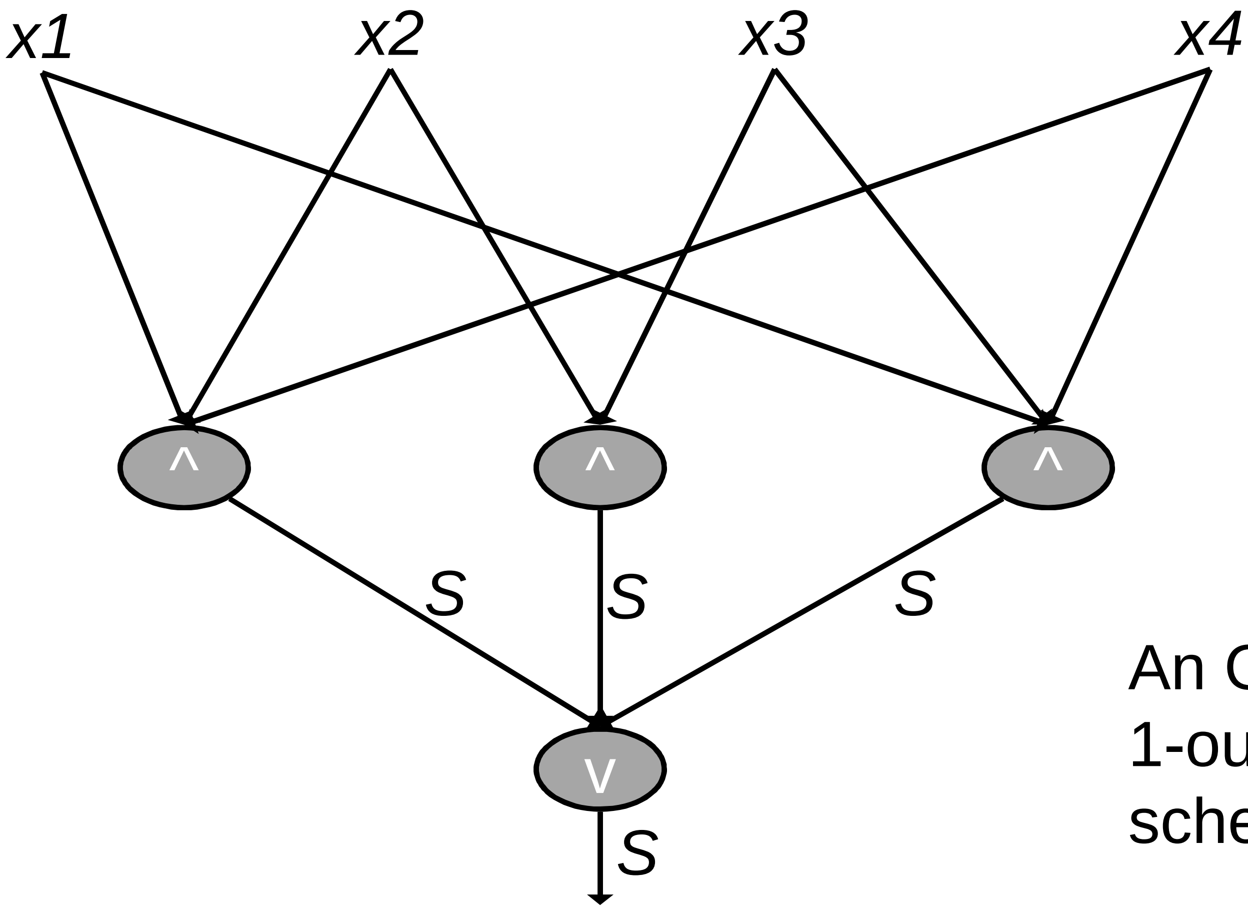
The monotone circuit construction (Benaloh-Leichter)

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



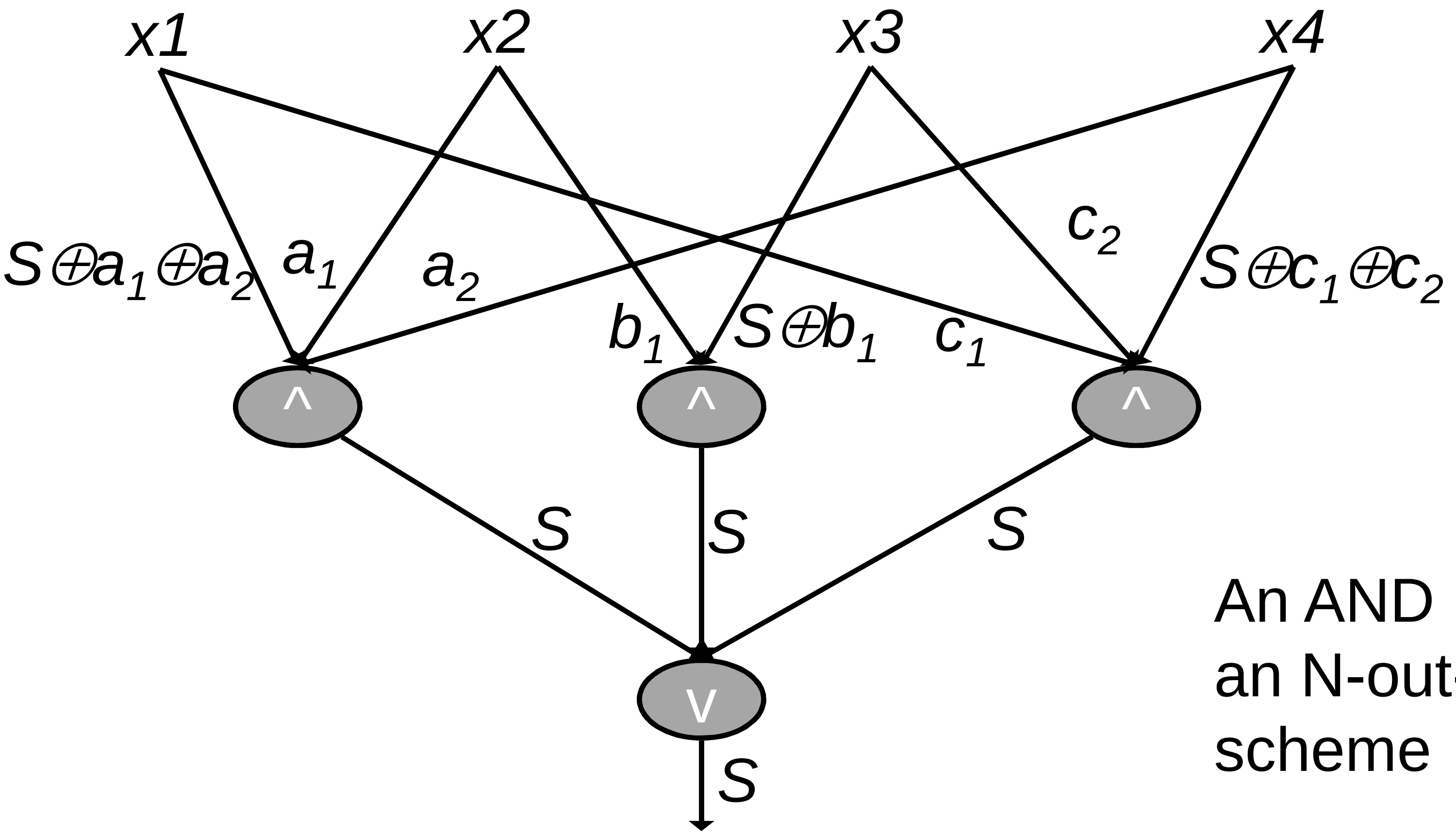
Handling OR gates

Starting from the output gate and going backwards



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

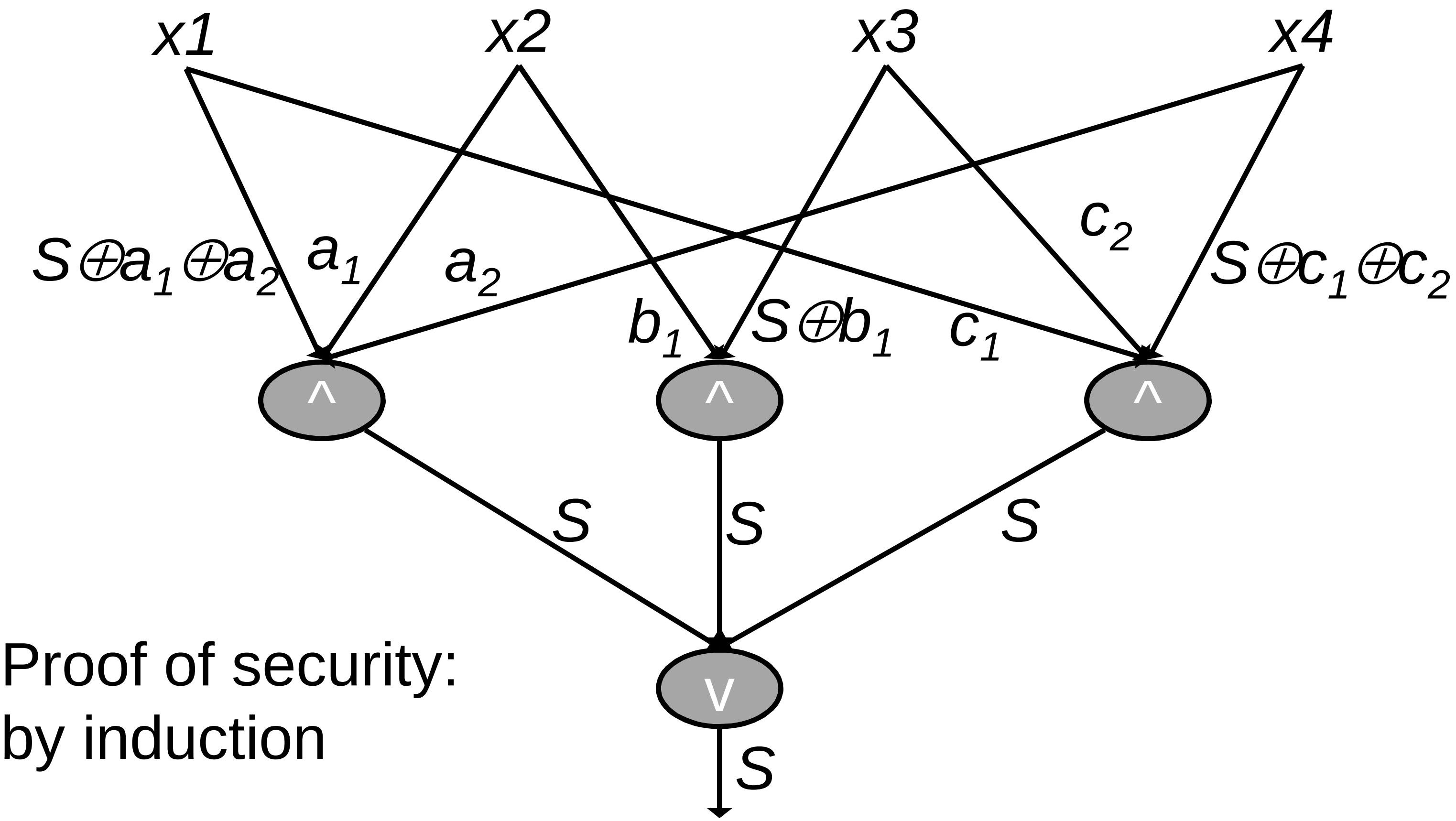
Handling AND gates



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

Handling AND gates

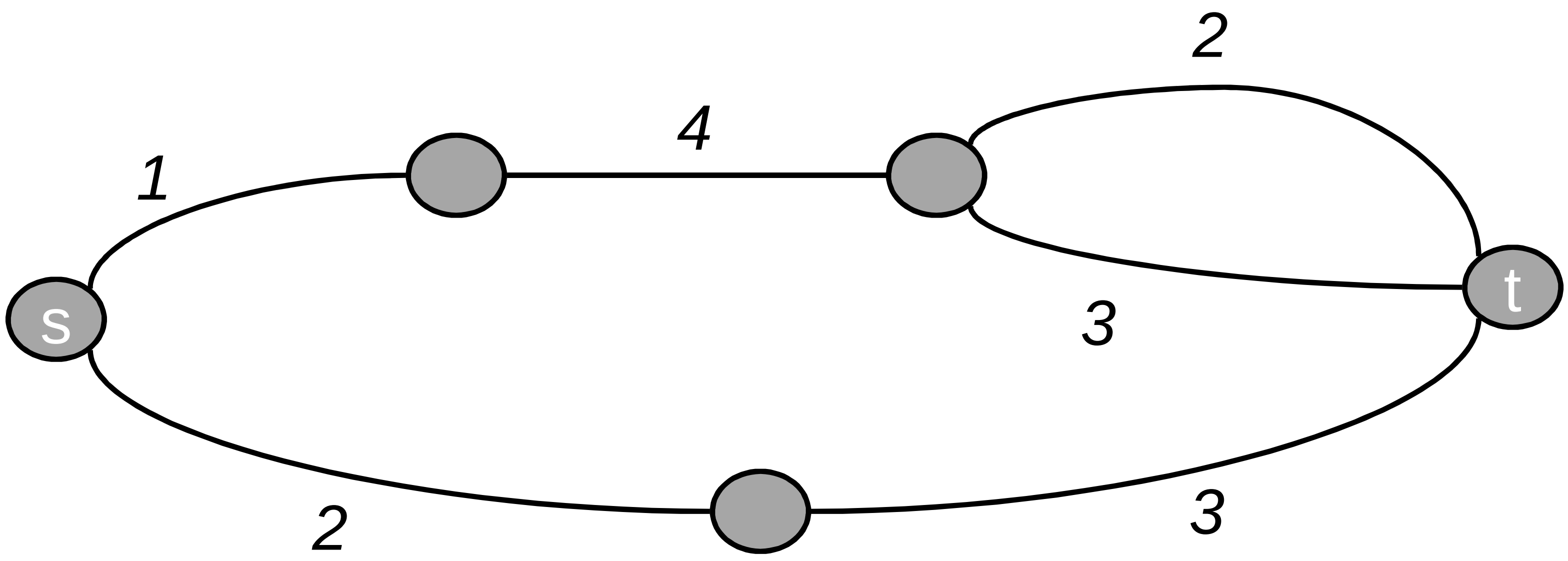
Final step: each user gets the keys of the wires going out from its variable



Proof of security: by induction

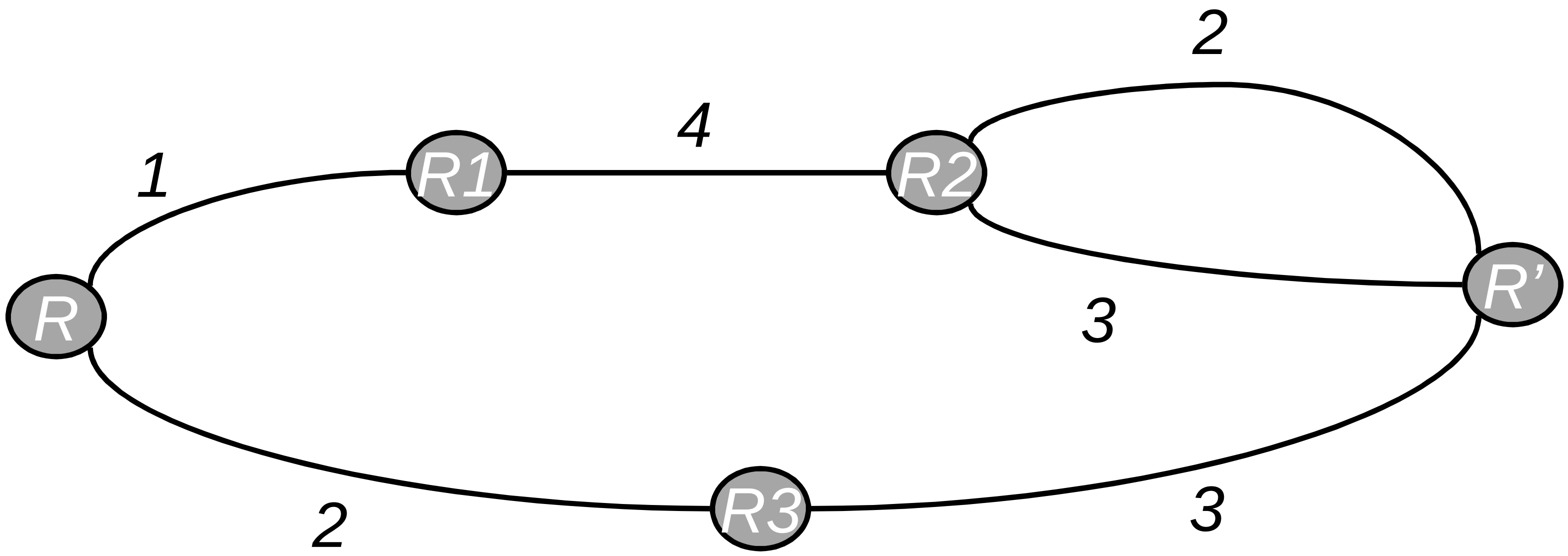
A graph based construction

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

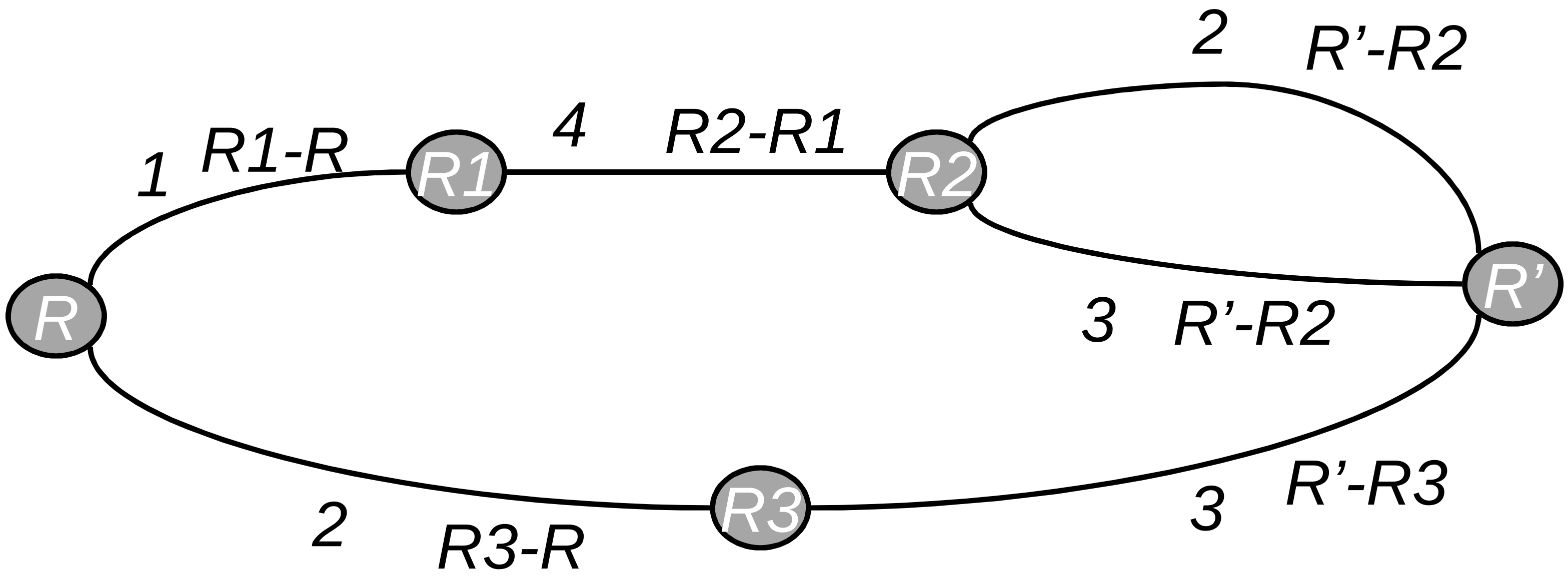


A graph based construction

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?