Proof of knowledge

From Wikipedia, the free encyclopedia

In cryptography, a **proof of knowledge** is an interactive proof in which the prover succeeds 'convincing' a verifier that it knows something. What it means for a machine to 'know something' is defined in terms of computation. A machine 'knows something', if this something can be computed, given the machine as an input. As the program of the prover does not necessarily spit out the knowledge itself (as is the case for zero-knowledge proofs[1]) a machine with a different program, called the knowledge extractor is introduced to capture this idea. We are mostly interested in what can be proven by polynomial time bounded machines. In this case the set of knowledge elements is limited to a set of witnesses of some language in NP.

Let x be a language element of language L in NP, and $W(x)$ the set of witnesses for x that should be accepted in the proof. This allows us to define the following relation: $R = \{(x, w) : x \in L, w \in W(x)\}$.

A proof of knowledge for relation R with knowledge error κ is a two party protocol with a prover P and a verifier V with the following two properties:

1. **Completeness**: if $(x, w) \in R$, the prover P who knows witness w for x succeeds in convincing the verifier V of his knowledge. More formally: $Pr(P(x, w) \leftrightarrow V(x) \rightarrow 1) = 1$
2. **Validity**: Validity requires that the success probability of a knowledge extractor E in extracting the witness, given oracle access to a possibly malicious prover $\tilde{P}$, must be at least as high as the success probability of the prover $\tilde{P}$ in convincing the verifier. This Property guarantees that no prover that doesn't know the witness can succeed in convincing the verifier.

Contents

Details on the definition

This is a more rigorous definition of **Validity**:[2]

Let R be a witness relation, $W(x)$ the set of all witnesses for public value x, and κ the knowledge error. A proof of knowledge is κ-valid if there exists a polynomial-time machine E, given oracle access to $\tilde{P}$, such that for every $\tilde{P}$, it is the case that $E^{\tilde{P}(x)}(x) \in W(x) \cup \{\perp\}$ and

$$\Pr(E^{\tilde{P}(x)}(x) \in W(x)) \geq \Pr(\tilde{P}(x) \leftrightarrow V(x) \rightarrow 1) - \kappa(x).$$

The result $\perp$ signifies that the Turing machine E did not come to a conclusion.

The knowledge error $\kappa(x)$ denotes the probability that the verifier V might accept x, even though the prover does in fact not know a witness w. The knowledge extractor E is used to express what is meant by the knowledge of a Turing machine. If E can extract w from $\tilde{P}$, we say that $\tilde{P}$ knows the value of w.

This definition of the validity property is a combination of the validity and strong validity properties in [2]. For small knowledge errors $\kappa(x)$, such as e.g. 2^{-80} or $1 / \mathrm{poly}(|x|)$ it can be seen as being stronger than the **soundness** of ordinary interactive proofs.

Relation to general interactive proofs

In order to define a specific proof of knowledge, one need not only define the language, but also the witnesses the verifier should know. In some cases proving membership in a language may be easy, while computing a specific witness may be hard. This is best explained using an example:

Let $\langle g \rangle$ be a cyclic group with generator g in which solving the discrete logarithm problem is believed to be hard. The deciding membership of the language $L = \{x \mid g^w = x\}$ is trivial, as every x is in $\langle g \rangle$. However, finding the witness w such that $g^w = x$ holds corresponds to solving the discrete logarithm problem.

Protocols

Schnorr protocol

One of the simplest and frequently used proofs of knowledge, the *proof of knowledge of a discrete logarithm*, is due to Schnorr.[3] The protocol is defined for a cyclic group G_q of order q with generator g.

In order to prove knowledge of $x = \log_g y$, the prover interacts with the verifier as follows:

1. In the first round the prover commits herself to randomness r; therefore the first message $t = g^r$ is also called *commitment*.
2. The verifier replies with a *challenge* c chosen at random.
3. After receiving c, the prover sends the third and last message (the *response*) $s = r + cx$.

The verifier accepts, if $g^s = ty^c$.

Sigma protocols

Protocols which have the above three move structure: commitment, challenge and response, are called sigma protocols. The Greek Σ visualizes the flow of the protocol. Sigma protocols exist for proving various statements, such as those pertaining to discrete logarithms. Using these proofs, the prover can not only prove the knowledge of the discrete logarithm but also that the discrete logarithm is of a specific form. For instance it is possible to prove that two logarithms of y_1 and y_2 with respect to bases g_1 and g_2 are equal or fulfill some other linear relation. For *a* and *b* elements of Z_q, we say that the prover proves knowledge of x_1 and x_2 such that $y_1 = g_1^{x_1} \wedge y_2 = g_2^{x_2}$ and $x_2 = ax_1 + b$. Equality corresponds to the special case where *a* = 1 and *b* = 0. As x_2 can be trivially computed from x_1 this is equivalent to proving knowledge of an *x* such that $y_1 = g_1^x \wedge y_2 = (g_2^a)^x g_2^b$.

This is the intuition behind the following notation, which is commonly used to express what exactly is proven by a proof of knowledge.

$$PK\{(x) : y_1 = g_1^x \wedge y_2 = (g_2^a)^x g_2^b\},$$

states that the prover knows an *x* that fulfills the relation above.

Applications

Proofs of knowledge are useful tool for the construction of identification protocols, and in their non-interactive variant, signature schemes. Such schemes are:

- Schnorr signature

They are also used in the construction of group signature and anonymous digital credential systems.

See also

- Cryptographic protocol
- Zero-knowledge proof
- interactive proof system
- Topics in cryptography
- Zero-knowledge password proof
- Soundness (interactive proof)

References

1. ^ Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (http://portal.acm.org/citation.cfm?id=63434) . *Proceedings of 17th Symposium on the Theory of Computation*, Providence, Rhode Island. 1985. Draft. Extended abstract (http://theory.lcs.mit.edu/~cis/pubs/shafi/1985-stoc.pdf)
2. ^ ***a b*** Mihir Bellare, Oded Goldreich: On Defining Proofs of Knowledge (http://www-cse.ucsd.edu/~mihir/papers/pok.ps) . CRYPTO 1992: 390–420
3. ^ C P Schnorr, Efficient identification and signatures for smart cards, in G Brassard, ed. Advances in Cryptology – Crypto '89, 239–252, Springer-Verlag, 1990. Lecture Notes in Computer Science, nr 435

External references

- Helger Lipmaa's cryptology pointers (http://research.cyber.ee/~lipmaa/crypto/link/zeroknowledge/pok.php)