

Hash Tube Signature Scheme

Satoshi Nakamoto
satoshin@gmx.com
25.12.2015

Abstract. A Hash Tube signature scheme is proposed, with an explanation how the individual tube levels are built. The signature and verification procedures on this virtual object are introduced. This scheme is applied to the signing of a cryptographic payments. Methods how the approach mitigates the double spend risk are evaluated in a practical retail-like use cases.

Keywords: transaction signature, double-spend, p2p electronic cash, retail, applied cryptography

1. Introduction

Before the advent of the public key cryptography, several methods to produce a signature schemes have been explored. In this paper, we first focus mainly on one-time signature schemes, and in particular those that rely on the properties of a class of functions known as one-way (or hash) functions. We revisit major achievements in the area: Lamport, Weierstrass and a Merkle tree based approach. We describe the important properties that a hash function must satisfy in order to be suitable for use in this scheme. Later, we focus on the application of these one-time signatures for signing transactions in a retail scenario, the best practices that have to be followed, and the guarantees these best practices offer for an acceptance of a 0-confirmed and 1-confirmed transactions, and describe how the counterparty risk shifts from a hostile double-spending customer to a miner who is in this case capable to sweep the hostile user's funds (not just the fee) as a mining profit. In a distant future when the subsidy drops, this "caught double-spend cash", together with fees, may form a substantial fraction of the total mining rewards. Partnerships between miners and merchants can enable merchants to fully reclaim these potential double-spend losses if they do occur.

2. Previous approaches

Theoretical advances brought us the surprising result, that given an one-way function f , a signature scheme can be constructed that relies only on the onewayness of this primitive f . Major approaches to deploy this result to production use are these:

Lamport signatures

Conceptually, every bit of the message is signed by revealing the preimage from a previously agreed upon collection of images. Man in the middle may interfere by hiding the revealed values, this is solved by having two distinct images in a row (left, right) for the 0-bit and 1-bit.

Weierstrass signatures

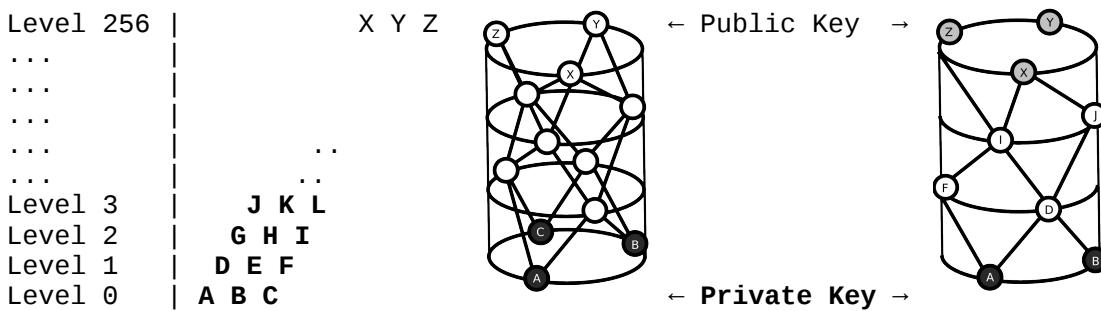
Turns out when we hash the preimage multiple times, we gain smaller signatures at the expense of verification time. This is similar to the exponentiation in ECDSA. The message is semantically represented as if encoded in unary, each unary digit is encoded as yet another round of hash function applied on a previous result. Again, man in the middle may hide some rounds, thereby making the unary value smaller. The obvious fix is to have two separate private keys producing two conceptual unary values with opposite signs. The first one represents an upper bound of a positive value of message, the second one the lower bound. From this follows that since the unary signature value is not larger and not smaller, so it must be equal to the message value.

Merkle-tree signatures

The most significant procedure here is the private key generation. The private keys are the exponentially numerous leafs of the Merkle tree. Upon completion, a key root value is found that becomes the corresponding public key. Each bit of the message is signed by a move of the cursor from parent to a child. 0-bit is encoded by moving to the left child, 1-bit is encoded by moving to the right child. Before each move the opposite child is pushed to the signature. At the end, the matching child is revealed. This scheme can be trivially reused, producing smaller signatures on further runs due to redundancy. Perhaps if the bytes of the message are unique and sorted, this scheme seems practical.

3. Hash tube construction

Each level consists of three values. A hash tube root is a triple of a random numbers (a, b, c). This root is denoted level 0. Variables on a next level are computed using the one way compression function on a nearby values in the current level:



The topmost values x,y,z become the public key. Now, a n-bit message can be signed with a tube having n- levels using a standard algorithm adapted from the Merkle tree signature algorithm, with a wrap around. At level 0 A is always pushed to the signature. Depending on the first bit, B or C is pushed to the signature. The cursor now moves to level 1 and determines what to push based on the second bit and so on.

When signing a message digest this way, a 8KB signature is produced. The verification is straightforward. First, an A is the fixed node on level 0, popped from the first value of the sig. Second value from the sig is popped, and hashed to A from left side or from the right side depending on the first bit of the message. The result is the fixed node on level 1, to which again sig value is popped and hashed from the side determined by the second bit of the message. This is repeated 256 times, yielding one of the values of x,y,z, or something else if the signature or the message is tampered with.

4. Properties of a suitable one-way function

Informally, the required function f is an one-way compression function that is known to be suitable for the construction of a Merkle trees. Let $f : \{0, 1\}^{2n} \rightarrow \{0, 1\}^n$ be a deterministic compression function. There is $2^{n \times 4^n}$ possible functions. A block cipher for example cannot be used as f due to the existence of the decrypt operator.

Now, let $g(a,b,c) \rightarrow i,j,k$ be the one-way function that calculates the next level of the tube. $i = f(a,b); j = f(b,c); k = f(c,a);$

1. Before signing, g^n must be **preimage resistant**. This is the single most critical property in the crypto currency application. A breach of this will allow an unauthorized user to spend someone else's funds.
2. After signing, g^n must be **preimage resistant given % of the preimage** (given values A,B or A,C)
3. To make the private key hard to crack from a signature, desired function f must be not only preimage resistant, but the f must be **preimage resistant given the half or the preimage**. However, cracking some possible private key value producing the known level 1 hash (D or F) from given values A,B or A,C does not imply the collapse of the scheme, because the resulting public key will perhaps not match the original x,y,z.