Mercurial Signatures.

A canonical digital signature scheme consists of three algorithms: key generation, signing, and verifying. It needs to be (1) correct: verification accepts $(PK, M, \sigma)$ if $\sigma$ is the output of the signing algorithm on input $(SK, M)$, for $SK$ corresponding to $PK$, and (2) unforgeable: (informally) a signature that verifies under $PK$ can only be produced by $PK$’s owner.

In a mercurial signature, public keys and messages are partitioned into equivalence classes using relations $\equiv_k$ and $\equiv_m$, and there are additional algorithms:

* ConvertSig: On input $(PK, M, \sigma)$ where $\sigma$ is a valid signature on $M$ under public key $PK$, output $(PK', M, \sigma')$, where $PK' \equiv_k PK$, and $\sigma'$ is a valid signature on $M$ under public key $PK'$.

* ChangeRep: On input $(PK, M, \sigma)$ where $\sigma$ is a valid signature on $M$ under public key $PK$, output $(PK, M', \sigma')$, where $M' \equiv_m M$, and $\sigma'$ is a valid signature on $M'$ under public key $PK$.

Further, for an appropriate choice of message space and public key space, the new message $M'$ cannot be linked to the original $M$, and the new public key $PK'$ cannot be linked to $PK$.

In this talk we will go over constructions and applications of mercurial signatures and open problems related to them. (Received September 15, 2020)