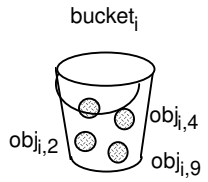

$$s_i = s_{i,2,1} \oplus \cdots \oplus s_{i,2,d}$$
$$s_i = s_{i,4,1} \oplus \cdots \oplus s_{i,4,d}$$
$$s_i = s_{i,9,1} \oplus \cdots \oplus s_{i,9,d}$$

Figure 4: Example where all traitors have different objects from bucket i: The three traitors can reconstruct s_i by three independent decompositions, and thus they must use all the second-level keys from one of them for this bucket.

5 Outlook

We have presented asymmetric fingerprinting and traitor-tracing schemes tolerating collusions of a significant number of traitors, the same numbers as the best currently known symmetric schemes tolerate. We have constructed and proved fairly general versions using secure 2-party protocols and sketched very efficient implementations of the concrete 2-party protocols, using significantly fewer commitments than the input length (not the circuit size!) of the underlying computation. Nevertheless the complexity of our constructions may still be prohibitive for realistic applications: The codes for the fingerprinting schemes are longer by a large constant factor than the underlying code from [BS95], which is already quite long. And at least one multiplication per input bit is required in all the schemes. Implementations that can use symmetric cryptologic primitives like hash functions on the bulk of the data are yet to be invented.

Acknowledgments

We are happy to thank *Matthias Schunter* and *Phil Janson* for interesting discussions and comments, the *DFG* (German Research Foundation) for financial support, and the *Isaac Newton Institute* in Cambridge, UK, for its hospitality to the first author while large parts of this paper were written.

References

[B87] Josh Cohen Benaloh: Verifiable Secret-Ballot Elections; Ph. D. Thesis, Yale University, September 1987.

[BCC88] Gilles Brassard, David Chaum, Claude Crépeau: Minimum Disclosure Proofs of Knowledge; Journal of Computer and System Sciences 37 (1988) 156-189.

[BM97] Ingrid Biehl, Bernd Meyer: Protocols for Collusion-Secure Asymmetric Fingerprinting; accepted for 14th Symposium on Theoretical Aspects of Computer Science (STACS) 1997.

[BMP86] G. R. Blakley, Catherine Meadows, G. B. Purdy: Fingerprinting Long Forgiving Messages; Crypto '85, LNCS 218, Springer-Verlag, Berlin 1986, 180-189.

[BRD95] F. M. Boland, J. J. K. Ó Ruanaidh, C. Dautzenberg: Watermarking Digital Images for Copyright Protection; 5th IEE International Conference on Image Processing and its Applications, Edinburgh, 1995, 326-330.

[BS95] Dan Boneh, James Shaw: Collusion-Secure Fingerprinting for Digital Data; Crypto '95, LNCS 963, Springer-Verlag, Berlin 1995, 452-465.

[C95] Germano Caronni: Assuring Ownership Rights for Digital Images; Proceedings VIS '95; Vieweg, Wiesbaden 1995, 251-263.

[CDG88] David Chaum, Ivan B. Damgård, Jeroen van de Graaf: Multiparty Computations ensuring privacy of each party's input and correctness of the result; Crypto '87, LNCS 293, Springer-Verlag, Berlin 1988, 87-119.

[CFN94] Benny Chor, Amos Fiat, Moni Naor: Tracing traitors; Crypto '94, LNCS 839, Springer-Verlag, Berlin 1994, 257-270.

[CKLS96] Ingemar J. Cox, Joe Kilian, Tom Leighton, Talal Shamoon: A Secure, Robust Watermark for Multimedia; Information Hiding Workshop, University of Cambridge, UK, 30.5.-1.6.1996, Preproceedings, 175-190 (proceedings to appear in LNCS, Springer-Verlag).

[DH76] Whitfield Diffie, Martin E. Hellman: New Directions in Cryptography; IEEE Transactions on Information Theory 22/6 (1976) 644-654.

[GMR88] Shafi Goldwasser, Silvio Micali, Ronald L. Rivest: A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks; SIAM J. Comput. 17/2 (1988) 281-308.

[GMW87] Oded Goldreich, Silvio Micali, Avi Wigderson: How to play any mental game – or – a completeness theorem for protocols with honest majority; 19th Symposium on Theory of Computing (STOC) 1987, ACM, New York 1987, 218-229.

[P96] Birgit Pfitzmann: Trials of Traced Traitors; Information Hiding Workshop, University of Cambridge, UK, 30.5.-1.6.1996, Preproceedings, 43-57, proceedings to appear in LNCS, Springer-Verlag.

[PS96] Birgit Pfitzmann, Matthias Schunter: Asymmetric Fingerprinting; Eurocrypt '96, LNCS 1070, Springer-Verlag, Berlin 1996, 84-95.

[W83] Neal R. Wagner: Fingerprinting; 1983 Symposium on Security and Privacy, IEEE, April 25-27 1983, Oakland, California, 18-22.

[Y82] Andrew C. Yao: Protocols for Secure Computations; 23rd Symposium on Foundations of Computer Science (FOCS) 1982, IEEE Computer Society, 1982, 160-164.

[Y86] Andrew C. Yao: How to Generate and Exchange Secrets; 27th Symposium on Foundations of Computer Science (FOCS) 1986, IEEE Computer Society, 1986, 162-167.

[ZK95] Jian Zhao, Eckhard Koch: Embedding Robust Labels Into Images For Copyright Protection; Congress on Intellectual Property Rights for Specialized Information, Knowledge and New Technologies, R. Oldenbourg Verlag, München 1995.

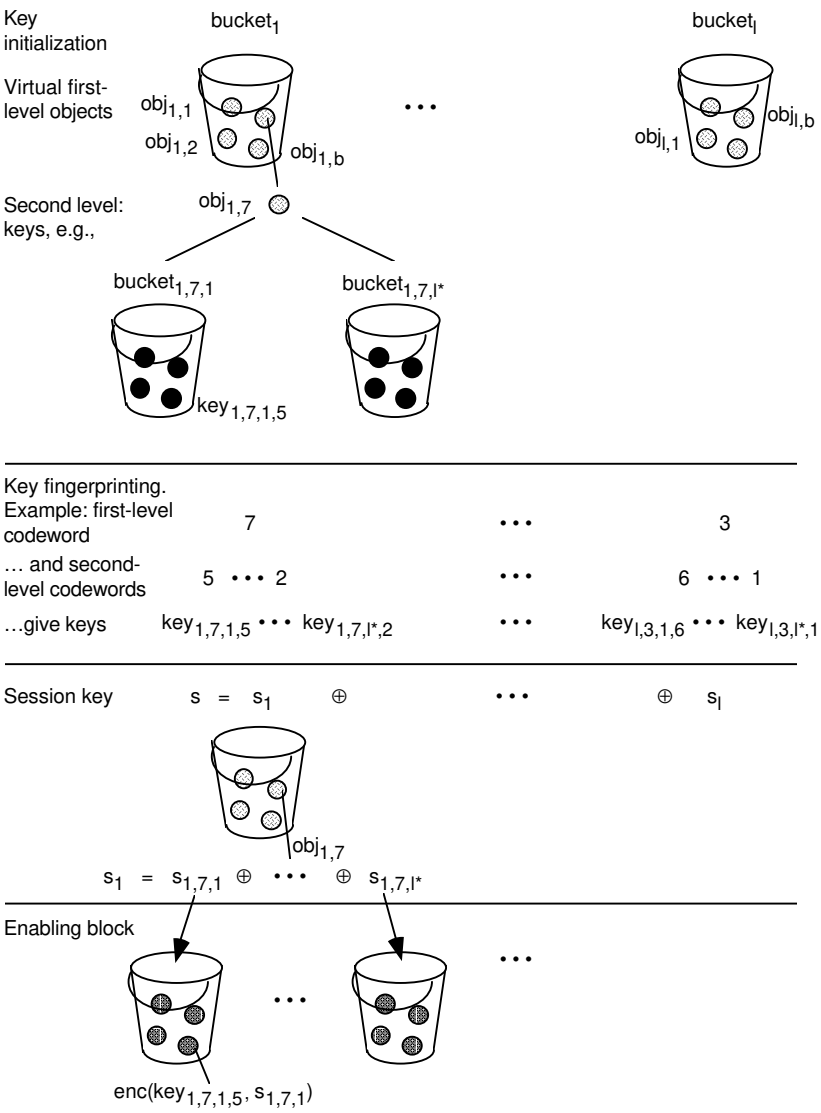

Figure 3: Two-level traitor-tracing scheme

4.4 Security for the Merchant

Without loss of generality, we assume that the traitors select exactly one second-level codeword $rid_{i,j}$ for each i and put the corresponding keys into the pirate decoder, because the merchant's choices in tracing cannot be worse.

It is clear by the construction, and if the 2-party protocol has been replaced by a trusted oracle, that the merchant will win a trial if one of these words $rid_{i,j}$ is the original one-way preimage rid_B of a traitor. Traitors can only choose among their own keys, and thus only among the symbols of their own corresponding second-level codewords. (Note that seeing prior trials does not help the traitors, because only codewords, and not keys, are shown there.) Hence $rid_{i,j}$ will in fact be an original rid_B if only one traitor had obtained $obj_{i,j}$ in her first-level codeword. The traitors can choose j, but if for any i, all traitors have obtained different objects $obj_{i,j}$, they cannot recombine any of their second-level codewords for this i, see Figure 4.

It remains to be shown that almost certainly, there is a bucket i where all the traitors' first-level codewords have a different symbol. The first-level codewords are chosen randomly by the merchant. Let $P_{coll,1}$ be the probability that a collision occurs for one particular bucket i, i.e., at least two traitors have the same $obj_{i,j}$. By the birthday problem, $P_{coll,1} \leq 1/2$ for our choice of $b \geq coll_size^2$. (One can easily see that this is a precise bound, and not only an asymptotic result.) Hence, the probability that there are collisions at all buckets is

$$P_{coll,l} = {P_{coll,1}}^{l} \leq (1/2)^{l} \leq 2^{-\sigma}.$$

decoder. During key fingerprinting, these partial codewords can therefore be hidden by a one-way function, instead of a commitment scheme; and the merchant can simply show to the judge that he has found the preimage of a one-way image of the buyer. We instantiate this idea based on the two-level schemes from [CFN94].

4.1 Symmetric Two-Level Schemes

The basic structure of the two-level traitor-tracing schemes in [CFN94] is shown in Figure 3. We describe the details immediately for the asymmetric construction.

4.2 Construction of the Asymmetric Two-Level Scheme

The following three cryptographic primitives must be fixed: a signature scheme, a one-way function f, and a protocol for secure 2-party computation.

As in all asymmetric schemes, we have parameters σ, *coll_size*, N (which is not needed here), and a computational security parameter k. From them, four technical parameters l, b, l^*, b^* are chosen, which denote the number and size of the first- and second-level buckets, respectively; see Figure 3. The choice is $l := \sigma$, $b := coll_size^2$, $l^* := k$ if k is the length of inputs to the one-way function f, and $b^* := 2$. The six subprotocols work as follows:

- Buyer key generation: Each buyer generates a key pair (sk_B, pk_B) of the given signature scheme and distributes her public key, pk_B, reliably.
- Key initialization is the same as in Figure 3, i.e., what is really chosen is l b l^* second-level buckets with b^* keys each.
- Key fingerprinting: The buyer B chooses a random "identity" rid_B of length k and represents it as a second-level codeword, i.e., as a word of length l^* over an alphabet of size b^*. (This is possible by the choice of l^* and b^*.)
  - The buyer computes the one-way image $im_B := f(rid_B)$ and sends it to the merchant.
  - The buyer signs the message $msg_B = (text, im_B)$ with her secret key sk_B, where *text* is a suitable text to make clear what this signature is about. The result is called sig_B.
  - The merchant verifies that sig_B is a correct signature on msg_B with the buyer's test key pk_B.

  Then the 2-party computation protocol is used for the following secret computation:
  - Inputs: The buyer secretly inputs rid_B. The merchant secretly inputs the keys from key initialization, a randomly chosen first-level codeword $word_B$ for this buyer, and (not necessarily secretly) the buyer's one-way image im_B.
  - It is verified that $f(rid_B) = im_B$. If not, the protocol is aborted.
  - The personal key corresponding to the first-level codeword $word_B$, and where all the second-level codewords are identical to rid_B, is selected and output to the buyer.
- Session sending works as in the symmetric scheme: Each session key s is broken up as the exor of l first-level shares $s_1, \ldots, s_l$. Each share s_i is broken up as the exor of l^* second-level shares in b independent ways, i.e., one decomposition per virtual object in a first-level bucket. Finally, each second-level share s_{i,j,i^*} is encrypted with each key from the corresponding second-level bucket. Now consider a buyer: She has one virtual object $obj_{i,j}$ from each first-level bucket i. This gives her access to the j-th decomposition of share s_i as follows: According to her second-level codeword, she has one key from each second-level bucket corresponding to $obj_{i,j}$, and thus she can decrypt each second-level share s_{i,j,i^*}. This allows her to reconstruct s_i, and finally the session key s.
- In tracing, the merchant has found certain indices of keys that are present in the pirate decoder. They yield one first-level word and one second-level word $rid_{i,j}$ for each i (i.e., the same structure of codewords as a buyer gets in Figure 3). He applies f to each of the l second-level words and compares the results with the stored values $im_B = f(rid_B)$. If he finds a match, he accuses the corresponding buyer of treachery, retrieves *text* and sig_B from the corresponding record, and uses $proof := (rid_{i,j}, sig_B)$.
- In a trial, the judge verifies that sig_B is a signature by the accused buyer on $msg := (text, f(rid_{i,j}))$.

As always, we exclude reactions on active attacks. However, no restrictions on the number of trials are necessary with this scheme.

Theorem 3. If all the underlying primitives are secure, the asymmetric traitor-tracing scheme constructed above is secure.

In this case, one can easily see that the merchant is protected from making wrong accusations, because the judge only makes verifications that the merchant has also made. The proof of Theorem 3 is given in the following two subsections.

4.3 Security for the Buyer

The security for the buyer primarily relies on the secrecy of rid_B, independent of the 2-level structure: To cheat a buyer, a merchant must first show a message $(text, im_B)$ with a valid signature by the accused buyer. If the buyer only signs messages with the appropriate texts within traitor tracing, the security of the signature scheme implies that this signature must have been made in key fingerprinting. Thus $im_B = f(rid_B)$ for a random value chosen by the buyer.

In this execution of key fingerprinting, the merchant obtains no output that depends on rid_B other than via $f(rid_B)$, if we assume that the 2-party protocol has been replaced by a trusted oracle. Moreover, we have excluded reactions on active attacks in session sending, and buyers do not do anything in trials. Thus the merchant never gains additional knowledge about rid_B. Hence, by the security of the one-way function (a fairly strong version), it is infeasible for him to compute a preimage of im_B, but such a preimage would be needed to win a trial against this buyer.

- Fingerprinting corresponds almost exactly to Section 2.3: The buyer chooses a random codeword $word_B$ of length l over the alphabet $\{1, \ldots, n\}$.
  - The buyer makes a commitment com_B to $word_B$ and sends it to the merchant. The information needed to open it is called $open_B$. If the 2-party protocol used starts with input commitments, then com_B is directly made as an input commitment for this protocol.
  - The buyer signs the message $msg_B = (text, com_B)$ with her secret key sk_B, where *text* is a suitable text to make clear what this signature is about. The result is called sig_B.
  - The merchant verifies that sig_B is a correct signature on msg_B with the buyer's test key pk_B.

  Then the 2-party computation protocol is used for the following secret computation:
  - Inputs: The buyer secretly inputs $word_B$ and $open_B$. The merchant secretly inputs the data to be fingerprinted, the positions of the marks from data initialization, a randomly chosen index set $Set_B \subseteq \{1, \ldots, l\}$ with $l/2$ elements, and (not necessarily secretly) the buyer's commitment com_B.
  - It is verified that the strings $open_B$ really open com_B to reveal $word_B$. If not, the protocol is aborted. It is also verified that Set_B has at most $l/2$ elements.
  - The symbols of $word_B$ at the positions in Set_B are selected and output to the merchant. They will be called $halfword_trace_B$, and the other half $halfword_evid_B$.
  - The individual symbols of $word_B$ are mapped into the corresponding codewords of $\Gamma_0(n, d)$. The resulting word, which corresponds to one from Γ', is encoded in the mark positions. This yields the fingerprinted data, which are output to the buyer.
- In tracing, the merchant has found a redistributed version of his data, and identified a binary word encoded in the marks.
  - The first step is as in Algorithm 2 of [BS95]: For each of the l positions in the outer code, he applies the underlying identification procedure of the inner code Γ_0 to identify a symbol y_i from the alphabet $\{1, \ldots, n\}$.
  - This yields $word_{found} := y_1 \ldots y_l$. Now he proceeds as in Section 2.3: He searches among all the pairs $(Set_B, halfword_trace_B)$ for one that has at least $l/(4coll_size)$ symbols in common with $word_{found}$, accuses the corresponding buyer of treachery, and retrieves $msg_B = (text, com_B)$ and sig_B from the corresponding record.
- In the trial, the merchant uses $proof = (msg_B, sig_B, word_{accuse})$, where $word_{accuse}$ consists of $halfword_trace_B$ at the positions in Set_B, and of the symbols from $word_{found}$ in the remaining positions. The judge first verifies the signature with the accused buyer's public key, pk_B.

  Now the merchant wins if $word_{accuse}$ has at least $l/2 + l/(16coll_size)$ symbols in common with $word_B$, the content of the commitment com_B. If this statement is false, the buyer disavows it by opening the commitment or in a zero-knowledge proof.

Similar considerations about repeated trials apply as above.

Efficient 2-party protocol for marking. Essentially, we can use Construction 2 of [PS96] as an efficient realization of the 2-party protocol used to encode a word in the mark positions. For this, it is simpler to make the commitments immediately on the word from Γ'. An efficient zero-knowledge proof that all inner symbols are codewords of $\Gamma_0(n, d)$ was also given in [PS96]. Outputting a halfword can be done as in Section 2.3. Furthermore, note that the construction works not only for marking schemes that use black-and-white pictures in their original pixel representation, but for any scheme where all the marks are disjoint subsets of the data coefficients in some representation, and where two versions of the data are given for each mark. For instance, this holds for [ZK95, BRD95]. Furthermore, higher residue commitments seem to be usable for a much larger number of bits than mentioned in [PS96], see [B87]. Once again, the result is extremely efficient for a 2-party protocol, significantly less than one commitment per input bit. Nevertheless, the computation and the code length itself may still be prohibitive for realistic data.

3.3 Security

For the security of this scheme, we need the Marking Assumption from [BS95], which says that the traitors cannot detect, and not even destroy without detection, any single mark in the data, unless their copies differ in this mark. This implies rather strong fault-tolerance measures for the actual representation of each single mark, but as all schemes in [BMP86, BS95] rely on this assumption, so will we for now.

Theorem 2. If the Marking Assumption holds and all the underlying primitives are secure, the asymmetric fingerprinting scheme constructed above is secure.

Security of the buyer. As in the proof of asymmetric traitor tracing, we can show that the merchant must guess at $halfword_evid_B$, and guess at least $l/(16coll_size)$ symbols correctly. As l and n in this construction equal l and b, respectively, in Section 2.3, the probability that he succeeds is again at most $2^{-\sigma}$.

Security of the merchant in tracing. The proof starts as in [BS95]: The security of the inner code Γ_0 guarantees that for each of the l positions of the outer code, a symbol y_i is identified that belongs to the codeword of one traitor (with an error probability of $2^{-\sigma-1}$ overall).

Now we are in the same situation as with traitor tracing: Without loss of generality, the traitors had to select $word_{found}$ of length l made up entirely of symbols from their own outer codewords. The probability that the merchant can identify one of them is therefore exactly as in asymmetric traitor tracing.

Probability that the merchant wins the trial. This, too, proceeds just as in traitor tracing on $word_{found}$.

4 Asymmetric Traitor Tracing with Two-Party Trials

In this section, we construct a traitor-tracing scheme where, technically, the buyer is not needed to carry out a fair trial. Thus there are no guesses by the merchant that the buyer has to disavow.

The basic idea is to fix partial codewords such that they can only be changed as a whole, and thus that they alone still identify a traitor. Hence, in spite of collusions, the merchant will find some of these partial codewords intact in a pirate

This probability equals the probability $P(Red \geq r^*)$ that a random sample of $l/2$ balls from a pool of l balls, among which at least $R := l/coll_size$ are red, contains at least $r^* := l/(4coll_size)$ red balls. Balls are not put back during the selection. As the same problem with other parameters occurs again in Section 2.6, we continue with arbitrary parameters R and r^*, where $r^* < R/2$ and $R < l/2$. The probability that the sample contains exactly r red balls is

$$P(Red = r) = \frac{\binom{R}{r}\binom{l-R}{l/2-r}}{\binom{l}{l/2}},$$

the density function of the hypergeometric distribution. We first show that this function decreases exponentially if we are far enough away from $R/2$:

$$\frac{P(Red = r-1)}{P(Red = r)} = \frac{\binom{R}{r-1}\binom{l-R}{l/2-r+1}}{\binom{R}{r}\binom{l-R}{l/2-r}} = \frac{r}{R-r+1}\,\frac{l-R-l/2+r}{l/2-r+1}$$

$$\leq \frac{r}{R-r+1} =: quot_R(r).$$

In the inequality, we used that $r < R/2$ and thus $R - r > r$. As $quot_R(r)$ decreases if r decreases, we can derive

$$P(Red = r-x) \leq \left(\frac{r}{R-r+1}\right)^x P(Red = r)$$

for any $x \leq r$, and then bound the entire head end of the distribution:

$$P(Red \leq r-x) \leq \left(\frac{r}{R-r+1}\right)^x P(Red \leq r),$$

because every term in the left-hand sum is bounded by a term in the right-hand sum. We now use $r = 3R/8$ and $x = R/8$, which yields our original $r^* = r - x = R/4$:

$$P(Red \leq R/4) \leq \left(\frac{3R/8}{5R/8+1}\right)^{R/8} P(Red \leq 3R/8)$$

$$\leq \left(\frac{3}{5}\right)^{R/8} \cdot 1 \leq 2^{-R/16} < 2^{-\sigma}.$$

For the last inequality, recall that $R = l/coll_size$ and $l \geq 64coll_size \cdot \sigma$. Thus we have shown that the traitor T fulfils the search criterion in tracing.

2.6 Probability that the Merchant Wins the Trial

Now we assume that the merchant has identified a buyer B and initiates a trial. (B will typically be the traitor T from Section 2.5, but others might fulfil the search criterion, too.) We show that the merchant will almost certainly win.

By the security of the underlying primitives, the buyer can only open her commitments (or prove in zero-knowledge that she can) to reveal the word $word_B$ that she had input to the 2-party protocol in key fingerprinting, and the $l/2$ symbols from $halfword_trace_B$ that the merchant uses in $word_{accuse}$ are correct. It remains to show that at least $l/(16coll_size)$ further symbols in $word_{accuse}$ are correct. For this, it suffices to show that the probability that there exists a buyer whose $word_B$ has at least $l/(4coll_size)$ symbols in common with $word_{found}$, but whose $halfword_evid_B$ does not even contain $l/(16coll_size)$ of these common symbols, is very small. We first consider one buyer B; the overall probability is at most N times as large. From Section 2.5, we know that no information about Set_B becomes known to the traitors before this trial against B, even if B is a traitor.

Thus it suffices to bound the probability that a random sample of $l/2$ balls from a pool of l balls, among which at least $l/(4coll_size)$ are red, contains at least $l/(16coll_size)$ red balls. This is the same problem as in Section 2.5 with l and $l/2$ as above, $R = l/(4coll_size)$, and $r^* = l/(16coll_size) = R/4$ as above, and therefore

$$P(Red \leq R/4) \leq 2^{-R/16} \leq 2^{-\sigma^*}$$

with $\sigma^* = \sigma + log_2(N)$. The overall probability that the merchant loses the trial is therefore at most $N \cdot 2^{-\sigma^*} = 2^{-\sigma}$.

3 Asymmetric Fingerprinting Based on Random Codes

The best scheme for collusion-tolerant fingerprinting in [BS95] (Section 4.1) also uses random codes as a building block. We can therefore make it asymmetric by applying the same basic idea as in Section 2. Almost the same construction has been discovered independently in parallel in [BM97].

3.1 The Underlying Symmetric Code

The code of choice in [BS95] is presented in and around Theorem 17. It has three technical parameters, l, n, and d, and is constructed from two components; such constructions are usually called concatenated codes:

- An "outer" code C whose codewords are uniformly and independently chosen words of length l over an alphabet of size n. There are N codewords, where N is the number of buyers.
- A specific deterministic "inner" code called $\Gamma_0(n, d)$, which contains n binary codewords of length $d(n-1)$ Å dn and is secure against collusions of any size, i.e., up to n. Note that Γ_0 is not used alone because the codewords would be too long if n were large.

The concatenated code is called $\Gamma'(l, N, n, d)$. It consists of N words; each of them is a word of C where the alphabet is represented by the codewords of Γ_0. Thus the length of each codeword is $l \cdot d \cdot n$. The codewords are kept secret from the buyers. For details and the security proof, see [BS95]. Actually, it is only important that the traitors do not know the codewords of the honest buyers, and the codewords can be generated on demand for each buyer as in Section 2.

3.2 Construction of the Asymmetric Scheme

As in Section 2, a signature scheme, a commitment scheme, and a protocol for secure 2-party computation are used, and the protocol has parameters σ, $coll_size$, N, and k.

From these parameters, the three technical parameters are chosen as follows: $l := 64coll_size \cdot (\sigma + log_2(N))$; $n := 48coll_size$, and $d := 2n^2(log(4nl)+\sigma)$. Here n plays a similar role as b in Section 2.3.

- Buyer key generation: Each buyer generates a key pair (sk_B, pk_B) of the given signature scheme and distributes her public key, pk_B, reliably.
- Data initialization: This subprotocol is carried out by the merchant for an item of data to be sold. The merchant selects the positions of the marks for these data.

underlying primitives) is given in the following three subsections.

Efficient 2-party protocol for key selection. We briefly sketch an efficient implementation of the 2-party protocol similar to Construction 2 in [PS96], based on quadratic residue bit commitments. The buyer chooses an instance of the commitment scheme, i.e., a number $n = pq$ where p and q are primes with $p \equiv q \equiv 3 \bmod 4$, and proves its correctness. This allows her to decode any commitment later. She commits to the symbols of $word_B$ in unary representation, i.e., the j-th symbol of the alphabet is represented as a string of zeros with a one in the j-th position. Let the commitments be $BC_{i,j}$, $i = 1, \ldots, l$ and $j = 1, \ldots, b$. She proves in zero-knowledge that each symbol is of the correct form. This can be done fairly efficiently, e.g., like the zero-knowledge proof for the code Γ_0 in [PS96]; note that $word_B$ even in this representation is short compared with the keys. Similarly, we can construct a fairly efficient protocol where the merchant obtains half of these symbols by blinding and permuting $word_B$, proving correctness of this transformation in zero-knowledge, and letting the buyer decode and send back the result. The merchant can unblind it and reverse the permutation.

Now, for each bucket i, the merchant encodes the first key, $key_{i,1}$, in the buyer's commitment instance as $BC_key_{i,1}$. Recall that the commitment to a string means the concatenation of commitments to the bits of the string. For all other keys $key_{i,j}$, he computes the set $Diff_{i,j}$ of positions where it differs from $key_{i,1}$. He can now compute a commitment to the key chosen by the buyer for bucket i by starting with the string commitment $BC_key_{i,1}$, and for all $j = 2, \ldots, b$, multiplying the bit commitment $BC_{i,j}$ in at all positions in $Diff_{i,j}$. Note that multiplying quadratic residue commitments means exoring the contents. Thus the contents are only changed by the one commitment $BC_{i,j}$ that encodes the one in the unary string, and this gives $key_{i,j}$. The buyer can decode the resulting commitment.

Note that this is extremely efficient for a 2-party protocol, although it may still be too much in practice given the length of the keys: Not even the whole input is committed to (only one key per bucket), sent in committed form, and decrypted. The merchant's additional computation on the whole input is about one multiplication per two bits.

2.4 Security for the Buyer

To cheat a buyer, a merchant must first show a message ($text$, com_B) with a valid signature by the accused buyer. If the buyer only signs messages with the appropriate texts within traitor tracing, the security of the signature scheme implies that this signature must have been made in key fingerprinting. Thus com_B is a commitment made by the buyer, and the buyer knows strings $open_B$ to open it to reveal a certain $word_B$.

Hence it is sufficient to show that it is infeasible for the merchant to construct a word $word_{accuse}$ that has at least $l/2 + l/(16coll_size)$ symbols in common with $word_B$.

We first show that this execution of key fingerprinting gave the merchant no knowledge about $halfword_evid_B$. We assume that the 2-party protocol has been replaced by a trusted oracle. The only output to the merchant that is not a function of com_B and values that were already known to him before the protocol is $halfword_trace_B$, which is exactly the other half of $word_B$. As we have assumed independent commitments on the different symbols of $word_B$, it is clear that the commitment does not significantly increase the merchant's knowledge about $halfword_evid_B$.

Next we investigate whether there are other sources of information for the merchant. Other executions of key fingerprinting with this buyer are independent of $word_B$, and so are protocol executions that follow from them. Thus there remain session sending and trials corresponding to this execution of key fingerprinting. In session sending, the buyer usually makes no outputs at all, but she might react on active attacks; therefore we excluded such reactions explicitly. Prior trials before the choice of $word_{accuse}$ have either been excluded, or the buyer only gave a zero-knowledge proof in them. In the latter case, the only knowledge the merchant gains is that $halfword_evid_B$ is not in the set of halfwords close enough to the corresponding half of $word_{accuse}$. As we show next, this set is an exponentially small fraction of all halfwords; hence a polynomial number of repetitions does not increase the merchant's knowledge significantly.

Hence, now we believe that the merchant has no significant information about $halfword_evid_B$. It remains to be shown that the probability of guessing a halfword that has at least $l/(16coll_size)$ symbols in common with the buyer's is small. Let S be the random variable denoting the number of symbols guessed correctly. The probability of guessing one symbol correctly is $p := 1/b$. The same version of the Chernoff bound as used in [CFN94] implies $P(S \geq \beta pl) < (e^{\beta-1}/\beta^{\beta})^{pl}$ for all $\beta \geq 1$. For $\beta = 3$, this implies $Pr(S \geq 3l/b) < 2^{-l/b}$ and thus, with our choice of $b = 48coll_size$ and $l \geq 64coll_size \cdot \sigma$,

$$Pr(S \geq l/(16coll_size)) < 2^{-\sigma}.$$

2.5 Probability that the Merchant Can Identify a Traitor

We can assume without loss of generality that the traitors select a word $word_{found}$ and put the corresponding keys into the pirate decoder, because the merchant's choices in tracing cannot be worse. In this section, we compute the probability that the merchant will at least be successful in tracing; the trial is considered in Section 2.6.

As there are at most $coll_size$ traitors, and they can only choose among their own keys, $word_{found}$ must contain at least $l/coll_size$ symbols from one traitor. We fix one such traitor T.[3] We have to compute the probability that the set Set_T chosen for T contains at least $l/(4coll_size)$ of these symbols that $word_{found}$ and $word_T$ have in common.

The choice of Set_T is random and independent of everything else. Furthermore, (except in a trial against the same T, which the merchant would have won, as we will show, and could simply repeat if more than one trial against the same traitor is made at all) information about Set_T could only become known to the traitors in key fingerprinting for T. There, however, nothing depending on Set_T is output to the buyer at all, if the 2-party protocol has been replaced by a random oracle.

Thus we only have to show that a random selection of $l/2$ symbols almost certainly contains at least $l/(4coll_size)$ of the $l/coll_size$ common symbols.

[3] Note that $word_{found}$ need not have $l/2coll_size$ symbols in common with any of the pairs (Set_B, $halfword_trace_B$) for these traitors as it would if Set_B were the same for all buyers. If the traitors knew Set_T used for this specific traitor T, they could even use keys from T solely in the other half.

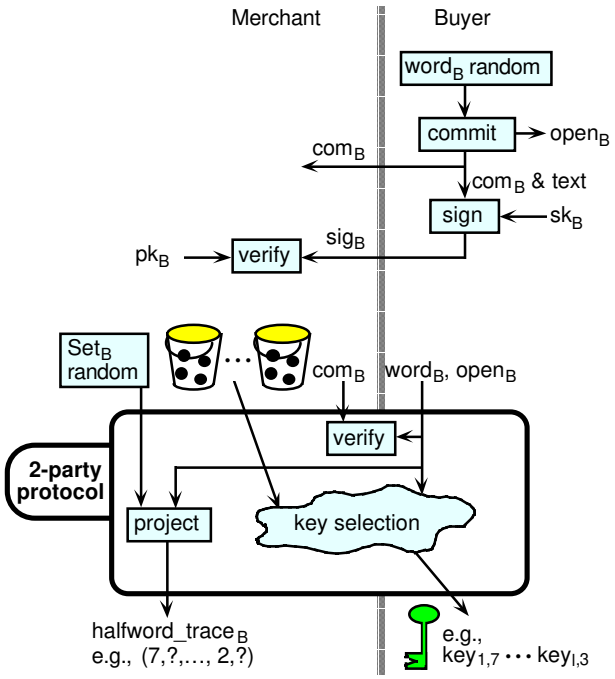

Figure 2: Two-party protocol in asymmetric traitor tracing with random codewords. The inputs *text* are not shown for readability.

- Key fingerprinting, see Figure 2: The buyer B chooses a random codeword $word_B$ of length l. (This choice was made by the merchant in the symmetric case.)
  - The buyer makes a commitment com_B to $word_B$ and sends it to the merchant. The information needed to open it is called $open_B$. If the 2-party protocol used starts with input commitments, then com_B is directly made as an input commitment for this protocol.
  - The buyer signs the message $msg_B = (text, com_B)$ with her secret key sk_B, where *text* is a suitable text to make clear what this signature is about. The result is called sig_B.
  - The merchant verifies that sig_B is a correct signature on msg_B with the buyer's test key pk_B.

  Then the 2-party computation protocol is used for the following secret computation.
  - Inputs: The buyer secretly inputs $word_B$ and $open_B$. The merchant secretly inputs the $l \cdot b$ secret keys from key initialization, a randomly chosen index set $Set_B \subseteq \{1, \ldots, l\}$ with $l/2$ elements[1], and (not necessarily secretly) the buyer's commitment com_B. Recall that in the case of a 2-party protocol starting with input commitments, inputting $open_B$ and com_B replace the first step of this protocol.
  - It is verified that the strings $open_B$ really open com_B to reveal $word_B$. If not, the protocol is aborted. Protocols with input commitments do this automatically. It is also verified that Set_B has at most $l/2$ elements.
  - The symbols of $word_B$ at the positions in Set_B are selected and output to the merchant. They will be called $halfword_trace_B$, and the other half $halfword_evid_B$.
  - The personal key corresponding to $word_B$ is selected and output to the buyer.

  The merchant's record is $record_M = (id_B, text, com_B, sig_B, Set_B, halfword_trace_B)$, and the buyer's record is $record_B = (text, word_B, open_B)$.
- Session sending is unchanged compared with Section 2.1.
- In tracing, the merchant has found certain indices of keys in the same way as in the symmetric scheme, and thus a word $word_{found}$ of length l. He searches among all the pairs $(Set_B, halfword_trace_B)$ for one that has at least $l/(4coll_size)$ symbols in common with $word_{found}$. (As in the figure, such a pair may be represented by writing $halfword_trace_B$ like a normal codeword with question marks on the positions not in Set_B.) He accuses the corresponding buyer of treachery, and retrieves $msg_B = (text, com_B)$ and sig_B from the corresponding record.
- In the trial, the merchant uses $proof = (msg_B, sig_B, word_{accuse})$, where $word_{accuse}$ consists of $halfword_trace_B$ at the positions in Set_B, and of the symbols from $word_{found}$ in the remaining positions. The judge first verifies the signature with the accused buyer's public key, pk_B.

  Now the meaning of the merchant's accusation is that $word_{accuse}$ has at least $l/2 + l/(16coll_size)$ symbols in common with $word_B$, the content of the commitment com_B. If this statement is false, the buyer disavows it by opening the commitment or in a zero-knowledge proof.[2]

As in previous schemes, we have to assume that buyers do not allow arbitrary active attacks on them. This means that for their own security, too, they must not show the decrypted data to anybody else, nor react if they seem to receive wrong enabling blocks. For instance, within the validity of one personal key, such as during one movie, they should only complain about too many wrong enabling blocks once, and without indicating which blocks were wrong.

Theorem 1. If all the underlying primitives are secure, the asymmetric traitor-tracing scheme constructed above is secure.

Additionally, the proof shows that the merchant is protected from damaging his reputation by making wrong accusations. This is an optional additional requirement on fingerprinting schemes; it means that even if there are more traitors than the scheme tolerates, the merchant will either fail in tracing or he will also win the trial (with the usual high probability). The proof (except for details like the error probabilities of the

1 Using a different Set_B with each buyer relieves the merchant from divulging information about a global secret *Set* in each trial. Instead, for traitor tracing (but probably not for fingerprinting), the merchants may simply decide that one trial per key initialization is enough. Moreover, in particular if trials are public, one has to consider in practice if seeing $word_{found}$ might be a temptation for new traitors, and thus multi-party protocols might be desired in trials, too. Then *Set* would not be divulged either and thus not need extra protection.

2 If the buyer has to open the commitment, she has to be protected from further accusations concerning this *text*. This may be done outside the cryptographic protocols, or by first giving the buyer a signature guaranteeing that $word_{accuse}$ is the only accusation made before any judges concerning this *text*.

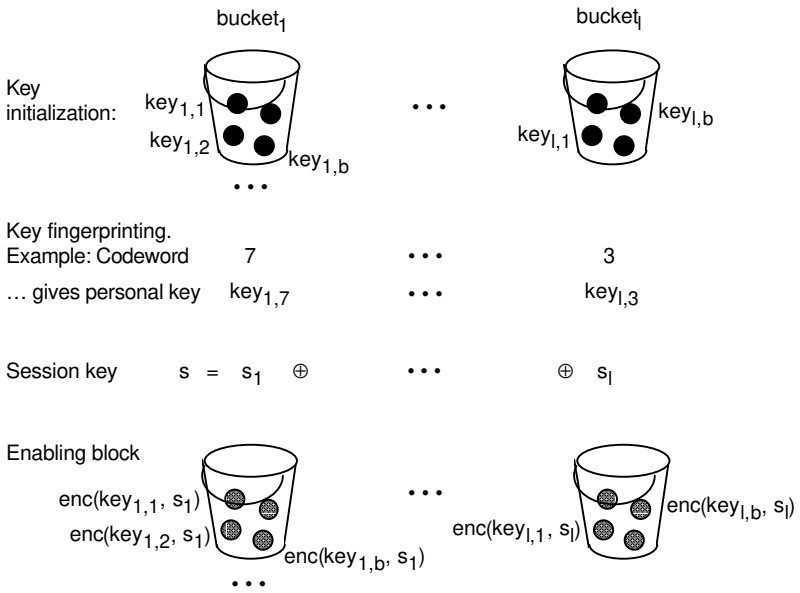

Figure 1: One-level traitor-tracing scheme

2.2 Cryptographic Primitives

For the following construction, three cryptographic primitives must be fixed: a digital signature scheme, a commitment scheme, and a protocol for secure 2-party computation. Note that a secure 2-party computation need not be inefficient if the function to be computed is simple; we will sketch an optimization for our particular problem after the main construction. We assume that the reader is familiar with signature schemes, first sketched in [DH76]; a precise definition is given in [GMR88]. However, we briefly sketch the other two primitives.

Commitments are the cryptographic analog of the following protocol: In the *commit* phase, one party puts some bits in a wooden box, locks the box, and sends it to the other party or puts it in a public place. Now the bits are hidden, but they can no longer be changed. In the *opening* phase, the party hands over the key of the box, so that the recipients can verify what bits the box contains. More technically, the commit phase has two outputs: the commitment, *com*, is an output for both parties, and the committing party additionally obtains a string *open*, which later enables her to carry out the opening phase. For several cryptographic realizations, see [BCC88]. For simplicity in the proof, we assume that commitments on different bits, or at least on different symbols in codewords, are made independently.

A protocol for secure 2-party computation achieves the following goal: Two parties have secret inputs x_1 and x_2, respectively. Both or one of them want to know $g(x_1, x_2)$, where g is a function known to both of them. However, Party 1 should not gain additional information about x_2, nor Party 2 about x_1. A nice example is the millionaires' problem from [Y82]: The parties are millionaires, and x_1 and x_2 denote how many millions they own. They want to find out who is richer, but without telling each other x_1 and x_2. Constructions have been given, e.g., in [Y86, CDG88, GMW87]. For security, we will assume that the 2-party protocol can replace a trusted oracle computing the same function. Most constructions of 2-party protocols start by requiring each party to commit to its inputs, and correctness of the output refers to the contents of these commitments.

2.3 Construction of the Asymmetric Scheme

The protocol is based on the following ideas: The buyer chooses her own codeword. The merchant learns part of this codeword in key fingerprinting, enough to identify the buyer if he later finds this buyer's key in a pirate decoder. The rest of the codeword is only given to the merchant in a commitment. Essentially, the merchant convinces a judge that he found a buyer's personal key by knowing the content of this commitment, because the content would otherwise be hidden. Complications in the protocol are due to two facts: The merchant usually does not find an entire personal key in a pirate decoder because of collusions; and he cannot prove on his own that he knows the content of the buyer's commitment, because he does not know the strings *open*.

The parameters of the protocol are σ, in which the error probability will decrease exponentially (as far as the error is not due to the underlying primitives), *coll_size*, the maximum number of colluding traitors, N, the maximum number of buyers to whom the data can be sold, and a parameter k for the computational security of the underlying primitives. For brevity, k is omitted in the following.

From these parameters, two technical parameters are chosen, $l := 64\,coll_size\cdot(\sigma + log_2(N))$ and $b := 48\,coll_size$. There are six subprotocols:

- Buyer key generation: Each buyer generates a key pair (sk_B, pk_B) of the given signature scheme and distributes her public key, pk_B, reliably.
- Key initialization is the same as in the symmetric scheme, i.e., the merchant chooses l buckets with b keys each.

in the restricted way of any cryptographic evidence, such as a digital signature. In particular, recall that if someone steals a buyer's copy out of the buyer's device together with any other secret data that may be there, no digital fingerprinting scheme can help distinguishing this situation from redistribution by the buyer.) A formal definition of asymmetric fingerprinting was given in [PS96]. Asymmetric traitor tracing has the same type of goals [P96].

Both articles [PS96, P96] contain constructions to transform a class of symmetric fingerprinting and traitor-tracing schemes, respectively, into asymmetric schemes. The main restriction of this class is that the schemes are memory-less, see [PS96, Definition 6]. More precisely, tracing may only rely on the initial information chosen for particular data to be sold and the merchant's random bits, but it must not depend on records taken during each purchase. However, the best schemes for traitor tracing in [CFN94] and for fingerprinting in [BS95] include the choice of a random codeword for each buyer, and for tracing, a list of pairs of the identity of a buyer and the corresponding codeword is searched. Hence these symmetric schemes cannot be used within the previous asymmetric constructions. The collusion tolerance of the asymmetric constructions was therefore quite restricted so far. There were two constructions without this restriction, both for asymmetric traitor tracing (Sections 4.1 and 4.3 in [P96]), but both had other restrictions instead: One would not fully count as broadcast encryption because of the amount of additional bandwidth a merchant needs, and in the other, the error probability in the security for the buyer is not negligible.

1.4 Overview of Our Results

The best symmetric schemes in the literature, both for fingerprinting and for traitor tracing, rely on random codes. In Sections 2 and 3 we show how to make such schemes asymmetric. Thus we obtain asymmetric schemes that have the same collusion tolerance as the best symmetric schemes. No new restrictions are made.

These schemes have 3-party trials, i.e., a fair dispute about whether a particular buyer is a traitor can only be carried out if this buyer can be found and asked to deny the charges. This does not only mean that the accused buyer is given an opportunity to defend herself, which should be clear in real life, but also that she has not forgotten the password to the device containing her secrets or died without leaving the password to her heirs etc. In Section 4, we show a traitor-tracing scheme with 2-party trials that also tolerates large collusion sizes.

2 Asymmetric Traitor Tracing Based on Random Codes

We start our constructions with a traitor-tracing scheme. First, we sketch the underlying symmetric traitor-tracing scheme.

2.1 The Underlying Symmetric Scheme

We use the so-called secret one-level scheme from [CFN94] with a small modification from [P96]. Its principles are shown in Figure 1.

The scheme has the parameter *coll_size*, the maximum number of traitors tolerated, and two technical parameters l and b, i.e., internal parameters to be determined from application-relevant parameters such as *coll_size* and the error probability. It has the following four subprotocols:

- Key initialization: This subprotocol is carried out by the merchant for an item of data to be sold, e.g., one movie in Pay-TV. The merchant chooses l sets of b keys of a symmetric encryption scheme. We call each set a bucket and denote the keys as shown in the top row of Figure 1.
- Key fingerprinting: For each buyer, the merchant independently and uniformly selects a codeword of length l over an alphabet of size b. This codeword designates one key from each bucket. The tuple of these keys is given to the buyer as her *personal key*. The merchant stores the pair of the buyer's identity and her codeword; this is what we call not memory-less.
- Session sending: Later, the merchant broadcasts the actual data encrypted in a normal way with session keys. Each session key is transmitted to the legitimate buyers in a preceding *enabling block* that is also broadcast. For this, the session key is broken up as the exor of l shares. Each share is encrypted with each key from the corresponding bucket. Now consider a buyer: As she has one key from each bucket, she can decrypt each share and thus recover the session key.
- Tracing: For tracing a traitor, it is assumed that at least one key from each bucket has been found in a pirate decoder. The merchant reduces this to exactly one key per bucket, e.g., the one with the smallest second index. Now he searches whether at least l/*coll_size* keys are present from any single buyer; if yes, he decides that this was a traitor. The underlying idea is that l keys are needed, and there are only *coll_size* traitors to provide them.

For the reasons why this symmetric scheme is secure, we refer to [CFN94, P96], but they will also become clear in the proof of our asymmetric scheme below.

Asymmetric Fingerprinting for Larger Collusions

Birgit Pfitzmann
Institut für Informatik, Universität Hildesheim
Samelsonplatz 1, D-31141 Hildesheim, Germany
<pfitzb@informatik.uni-hildesheim.de>

Michael Waidner
IBM Zürich Research Laboratory
Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland
<wmi@zurich.ibm.com>

Abstract. Fingerprinting schemes deter people from illegally redistributing digital data by enabling the original merchant of the data to identify the original buyer of a redistributed copy. So-called traitor-tracing schemes have the same goal for keys used to decrypt information that is broadcast in encrypted form. Recently, asymmetric fingerprinting and traitor-tracing schemes were introduced. Here, only the buyer knows the fingerprinted copy after a sale, and if the merchant finds this copy somewhere, he obtains a proof that he found the copy of this particular buyer. This gives both parties security in disputes. First constructions showed the validity of the concept.

However, these constructions did not yet achieve much security against collusions, in contrast to known symmetric schemes. This paper presents asymmetric constructions without this restriction.

1 Introduction

The main roles in a fingerprinting or traitor tracing scenario are:

- *merchants*, who sell digital *data*,
- *buyers*, who buy the data from these merchants,
- *traitors*, dishonest buyers who *redistribute* the data without the merchant's permission,
- *pirates*, who obtain these data from traitors,
- and *judges*, by which we mean any third parties called upon to settle disputes.

1.1 Fingerprinting

Fingerprinting is a class of mechanisms that support copyright protection without needing tamper-resistant hardware. As illegal redistribution cannot be prevented in this case, the goal is to deter people from it. In particular, the goal of fingerprinting is to deter the traitors, not the pirates, by allowing the merchant to identify the original buyer of a particular illegal "copy" of the data. This buyer can then be punished as a traitor.

The word copy was written in quotes above because the basic technique is that each buyer gets a slightly different version of the data. (Of course, if a pirate steals the copy of a buyer, the situation is indistinguishable from that where the buyer really is a traitor.) The differences introduced into the copies should neither decrease the usefulness of the data for the buyer, nor be detectable and thus removable by a traitor. For details of how this is typically achieved, see, e.g., [W83, C95, ZK95, BRD95, CKLS96].

For our purposes, we can concentrate on coding aspects that are rather independent of the type of data to be fingerprinted, as in [BMP86, BS95]. These papers assume that a method for embedding individual *marks* into data has been found such that each mark can be used to encode at least one bit. Thus a mark is a part of the data for which at least two versions exist. It is assumed that the only way for traitors to detect marks is to compare their individual versions of the data. The two papers investigated codes that maximize the collusion tolerance, i.e., how information can be encoded into the marks such that even a certain number *coll_size* of traitors cannot redistribute data without being traced.

1.2 Traitor Tracing

Traitor tracing, introduced in [CFN94], can be seen as a specific subclass of fingerprinting where, instead of real data, a key needed to decrypt the data is fingerprinted. The typical application is Pay-TV, where large amounts of data are broadcast in encrypted form, and the key to these data is sold. More generally, scenarios where encrypted data are broadcast and the keys are sold are called broadcast encryption.

The goal of traitor tracing, just like that of fingerprinting, is to deter traitors by making all the keys sold slightly different, so that the original buyer of such a key can be identified. Constructions were shown in [CFN94]; some are sketched below.

1.3 Asymmetric Fingerprinting and Traitor Tracing

In conventional fingerprinting schemes, both the buyer and the merchant know the data with the fingerprint. Thus, if a copy with this fingerprint turns up where it should not, one cannot assign responsibility to any one person unambiguously: The copy might not only have been redistributed by the buyer, but also by a dishonest employee of the merchant, or the merchant himself may want to gain money by wrongly claiming that there are illegal copies around. In other words, the merchant does not obtain means to prove to a judge that the buyer redistributed the copy.

Asymmetric fingerprinting schemes solve this problem in the following way: After a sale, *only* the buyer knows the data with the fingerprint. If the merchant later finds this copy somewhere, he can identify the buyer and prove to judges that he found the copy of this buyer. ("Prove" has to be understood

4th ACM Conf. on Computer and Communications Security, Zurich, April 2-4, 1997

© ACM 1997. Permission to make digital/hard copy of all or part of this material without fee is granted provided that copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication and its date appear, and notice is given that copyring is by permission of the Association for Computing Machinery, Inc. (ACM). To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.