Accountable Banking Transactions

Sebastian Mödersheim[1] and Siyu Chen[2]

Abstract: This paper shows how to apply the idea of *Three branches of Accountability* by Mödersheim and Cuellar to make banking transactions accountable, i.e., neither can the customer later deny to have placed the order, nor can the bank execute a transaction that the customer did not order. This is done in a general way that deliberately gives freedom to instantiate the system in several different ways, as long as it follows a few basic principles, and we show accountability holds in every instance.

Keywords: Accountability, Formal Verification, Security Protocols

1 Introduction

A security expert (whose identity we do not disclose here) once complained in a conversation with the authors about the following seemingly pointless security hurdle: after logging into his online banking system with a national Single-Sign On system (SSO), whenever he ordered a transfer, the bank would require another interaction with the SSO system. This nuisance, the expert argued, would only help if a user would leave open their machine unattended and this should better be prevented by automatic logout after short inactivity.

There is of course a different reason for the confirmation, and it is in fact very similar to the classic non-electronic banking. Here, a customer would in general need to show the bank clerk some identity document to authenticate themselves. Still, in order to make a bank transfer, the customer would have to fill out a form and *sign it*. The point of this signature is not authentication (the bank clerk already knows the identity of the customer), but gives the bank a proof that the customer indeed ordered this transaction. This proof is transferable, i.e., it can be shown to others. Thus, neither could a dishonest customer later deny a transaction they made, nor could a dishonest bank clerk (or dishonest bank) execute a transaction that had not been ordered without a substantial risk to themselves. In the electronic scenario, the second involvement of the SSO replaces the customer's signature, since typically the customers have no public/private key pairs with legal binding to their identity.

Accountability (see for instance [KTV10]) is a concept that protects security goals which cannot be enforced directly by cryptography, e.g., the goal that a customer never makes unfounded claims or a bank does not perform illegal transactions. Making the transactions accountable thus means that the actors cannot later deny what they did which can involve punishment in case of wrongdoing. We later discuss the relation with the common notion of *non-repudiation*.

1 DTU Compute, Kgs. Lyngby, Denmark, samo@dtu.dk, https://orcid.org/0000-0002-6901-8319
2 DTU Compute, Kgs. Lyngby, Denmark,

We use in this paper the framework *Three Branches of Accountability* by Mödersheim and Cuellar [MC22]. A main idea is here not to study a particular protocol, but define through a legislative framework which messages have legal meaning and what actions are illegal. Agents are thus free to do anything that is not forbidden, and the legislation does not prescribe the protocols to be used, e.g., whether a bank has to use TLS to secure the connection with the customers. This considerably departs from standard paths of security protocol design and verification. Normally, we have a set of honest agents that follow the protocol and an intruder who can act as a participant, but who does not necessarily follow the protocol. What the intruder can do is defined by the common Dolev-Yao model: the intruder controls the communication medium and can apply encryption and decryption functions with known keys. In our accountability model, *all* agents are like such intruders communicating over a network where everybody can add messages and everybody can see (but possibly not decrypt) all messages. This gives us a transition system in which agents can truly "do whatever they want". We will make a restriction on the agents behavior, but essentially the goal is to verify that this large transition system has no attack state, i.e., a state that violates given security goals.

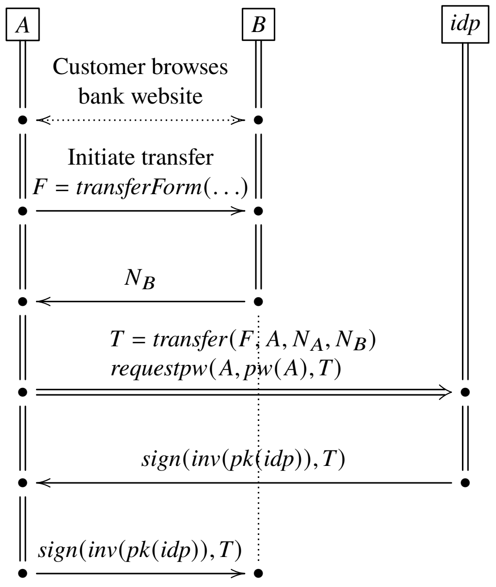
The legislative is the first branch of accountability, giving us a definition for every state in this transition system whether an agent has committed a crime (even though this is possibly not detectable for police and justice) and what legal terms hold in this state, e.g., that a public key PK is legally bound to agent A. The second branch is the executive branch which does not play any role in our case, and third is the judicial branch. It will be invoked in our case when a customer complains about a transaction, and should decide whether the customer or the bank is guilty of violating the law. The judicial branch is modeled as special transitions where an honest judge follows a protocol to interact with other participants. The participants can choose what to say to the judge (again within the Dolev-Yao model), and at the end of the protocol one or more participants are convicted.

There are now two things to prove: first, that this system is *lawful* in the sense that the innocent cannot be wrongly convicted, and that a violation of the security goal leads to some conviction.[3] We assume then that actors only commit a crime if it as *perfect* crime, i.e., where they are sure that they cannot be convicted. Under this assumption it then follows that the security goals are never violated.

The contributions of this paper are to formalize the accountability for banking transactions with SSO in a very generic way that can be instantiated by many concrete bank transaction systems and protocols. We then prove the lawfulness and accountability, i.e., violation of security goals leads to a conviction. We also show a minor variant that corresponds to a simple oversight where accountability would not hold.

3 The security goals in general do not require to prevent every illegal action: there may be illegal actions that the security goals are not directly concerned with (e.g., a user sharing their password with a friend). Moreover, we may have security goals that are not enforced by the three branches.

2 The Legislative



A
B
idp
Customer browses bank website
Initiate transfer $F = transferForm(\ldots)$
N_B
$T = transfer(F, A, N_A, N_B)$
$requestpw(A, pw(A), T)$
$sign(inv(pk(idp)), T)$
$sign(inv(pk(idp)), T)$

Fig. 1: Generic schema of a bank transaction between customer A, Bank B, and identity provider *idp*.

Figure 1 shows the generic schema of a transaction. It starts with a customer A who browses the website of their bank B, e.g., looking at their account balance. Typically, this entire communication is secured using a protocol like TLS and some authentication mechanism like a login with password. At some point A may decide to order a money transfer. For this purpose we assume there is an electronic form *transferForm* that contains a number of fields like the sender's and receiver's IBAN number, the amount (and currency) to be transfered. Such a form is some (non-cryptographic) way of structuring the information to be entered into the form, e.g., XML. Up to this point, we do not specify exactly how the interaction between A and B should work: this is up to the bank how to design their web interface, what authentication mechanism to use, and how to encrypt the transfer.[4] B will now generate a random number N_B and ask the user the authorize the transfer in an accountable way using N_B. To that end, since A does not have a legally registered public/private key pair, A turns to the identity provider *idp* in order to get the transfer signed by the *idp*. A first creates $T = transfer(F, A, N_A, N_B)$, another electronic form that contains the transfer form F, the name of A, a random number N_A created by A as well as N_B from before. A packs this into yet another form $requestpw(A, pw(A), T)$ that contains the name and password that A has with *idp*. This distinction of three forms is just for conceptual simplicity: the *transferForm*(...) contains the information that would also be present on a

4 In fact, for the purpose of the accountability proof below, the communication does not even have to be encrypted. However, for privacy this would be terrible: everybody on the network could observe A's interactions with the bank; also the authentication mechanism obviously cannot be password-based if messages were unencrypted.

non-electronic bank transfer form and is thus only banking-related without other technical aspects; the *transfer*(...) form is what should actually be signed by the *idp* containing the random numbers; and finally *requestpw*(...) is the message that A sends to the *idp* and that should not be observed by anybody else. This message indeed has to be encrypted in a way that only the *idp* can read, because it contains the user's password, and this will be part of the legal requirements below. We have in fact depicted this transmission in the figure using a double arrow to highlight that this transmission crucially needs confidentiality. Again, we do not require a particular way of achieving this: while one typically uses TLS, simply encryption with the public key of *idp* would be sufficient. If everything is fine, the *idp* signs the request and sends it back to the customer who can forward it to the bank, which then executes the transfer. Here $pk(idp)$ is the public key of *idp* and $inv(pk(idp))$ is the corresponding private key.

This protocol schema has some assumptions: the *idp* must be honest, otherwise there are trivial attacks against this. (In contrast, A and B may well be dishonest.) Further, nobody except A and *idp* knows $pw(A)$, and everybody knows that $pk(idp)$ is the public key of *idp* (and can thus verify the signatures produced here).

We call this a protocol schema because it leaves many details open and therefore gives banks the freedom to implement their online banking systems as they see fit, as long as they obey a few legal regulations. The three named formats *transferForm*, *transfer* and *requestpw* have distinct legal meaning, and thus the law must fix a particular way to implement these forms, e.g., as XML formats. Similarly, some cryptographic parameters must be fixed like permissable signature schemes and key sizes. We assume here that these details are fixed already.

The legal system now consists of the following laws; as is standard in legal text, some "commentary" will be given as footnotes:

§1 There is a public key legally bound to the identity provider, denoted $pk(idp)$ in the following. If any agent other than *idp* posses the corresponding private key $inv(pk(idp))$, then *idp* is punishable.[5]

§2 There is a set of actors who are registered as banks. To each bank B a public key denoted $pk(B)$ is legally bound. If any agent other than B posses the corresponding private key $inv(pk(B))$, then B is punishable.

§3 There is a set of actors who are registered as citizens, they each have a unique identifier and password registered with the *idp*. The password of citizen A is denoted $pw(A)$. If anybody other than A and the *idp* knows $pw(A)$, then A is punishable.[6]

5 As said before, we will assume that the *idp* is honest and therefore will never leak their private key, so they will never be punished.

6 The logistic requirements are fulfilled for instance in Denmark by the fact that every legal resident has a CPR number and authentication credentials with the national identity provider MitID. We do not consider here two-factor authentication for simplicity or mechanisms for changing the password for simplicity. The problem that an honest actor may lose their password is discussed below.

§4 When a client A wants to order a bank transfer, then they are required to generate a random number N_A and ask the bank for a random number N_B. Then they may issue the format $T = \mathit{transfer}(F, A, N_A, N_B)$ with F the fields of the transfer, and issue the format $\mathit{requestpw}(A, pw(A), T)$. Producing this message legally binds A to this transfer detailed by F, and A is punishable if N_A has been used in a different transfer. The client is also obliged to encrypt the message in such a way that only the *idp* can read it and that no agent can change the content of the message.[7]

§5 When the *idp* receives a message $\mathit{requestpw}(A, pw(A), T)$ that uses the correct password $pw(A)$ of A and T is a format *transfer* containing sender name A, then the *idp* may sign the message T with their private key $\mathit{inv}(pk(\mathit{idp}))$. If the *idp* signs with $\mathit{inv}(pk(\mathit{idp}))$ a *transfer* message other than according to this law, then the *idp* is punishable.[8]

§6 When a bank B receives *transfer* message T signed by the identity provider, it may execute the transfer requested in T, provided that: the random number N_B contained in T was freshly generated by B and has not been used in any other transaction, the name A corresponds to a legal account holder at B denoted by the IBAN number of the sender in T. A bank who performs a transfer without a signed transfer message T according to this law is punishable. The bank is obliged to save the signed T message and produce it when subpoenaed by a judge; if the bank fails to answer the subpoena to a transaction they have performed, they are also punishable.

§7 A customer can complain to a judge if a bank has performed a transfer without the customer having legally requested it. The client is punishable if they issue such a complaint while they did legally request that transfer.

3 The Judicial Branch

The second branch of The Three Branches of Accountability paper is the executive branch: the police who may discover some criminal activity, and provide the evidence to the judicial branch. This is not necessary in our case, so we directly come to the third branch, the judicial branch. This is about declaring how a judge should proceed when a customer A registers a complaint that a bank B made a transfer from A's account without A ordering that.

The judge has to follow a simple procedure: they subpoena the bank B to provide a signed transfer form that § 6 requires before the bank can make a transfer. If B does not produce a signed *transfer* form that matches the transaction, then B is convicted. Otherwise the customer A is convicted.

7 The requirement that no other agent than *idp* can learn it follows already from §3; here it also required that no other agent can alter the message. Note that the law leaves open how this is done, e.g., by public key encryption or TLS.

8 Note that the law only says *may*, because the *idp* shall not be punishable for instance for downtimes. Moreover the law does not forbid the *idp* to use its private key for other purposes, as long as the signed message is not of the *transfer* format.

Thus we have a procedure for the judge that will identify one participant as guilty, whenever a customer complains about a bank transfer they allegedly did not order. It may seem intuitively clear that this procedure is never convicting somebody who is innocent, i.e., somebody who abided the law. However, this is not the case as demonstrated by the following attack.

3.1 Breaking and Fixing the Transfer Protocol

Suppose an agent A issues a transfer request to the *idp* who signs it, and the client forwards this signed request to the bank B. Suppose now B is malicious and executes the exact same transfer two times, effectively doubling the amount that A transfers. Now A is intuitively right to complain about the second transfer (or alternatively about the first one). However, if they complain to the judge and the judge subpoenas B, B can show a valid signed request that matches the transaction, and thus, with the above judge procedure, the client would now be convicted without having broken any laws.

To fix the situation, recall that we have required that the transfer contains random numbers N_A and N_B, and that each side is obliged to create them freshly. Note that this is also subtle, because the bank cannot prevent that a nonce N_B they created could be used by a malicious customer in several different requests, so it is not so easy to tell in general who has failed to adhere to the protocol.

However, if the judge looks at two (or more) identical transfers that B has executed, and subpoenas for all of them, and B presents for each the *same* signed *transfer* form, including the same nonce N_B used in each of them, then B is actually punishable by § 6 which explicitly obliges B to check that N_B was generated by B and not used in another transaction. So even if for instance a malicious client has issued several transfers with the same N_B, it is the duty of B to check that each N_B can only be used once. It should also be noted that without the fresh random numbers N_A and N_B it would be impossible now to tell whether the bank or the client broke the law.

We thus modify both § 7 and the procedure of the judge (the change is underlined):

§ 7 A customer can complain to a judge if a bank has performed a transfer without the customer having legally requested it, <u>or more often than the client has requested it</u>. The client is punishable if they issue such a complaint while they did legally request that transfer <u>at least as many times as the bank executed it</u>.

The judge now follows this procedure: given a set of transfers that a client complains about, the judge subpoenas the bank B to provide signed *transfer* form for each transfer according to § 6. If B fails to produce a signed *transfer* forms with distinct values of N_B, then B is punishable. Otherwise the client is punishable.

4 Security Goals and Proof

4.1 Lawfulness

First, we prove that this system is *lawful*, i.e., that no innocent (i.e., law-abiding) agent can ever be convicted (which actually could happen in the first version as demonstrated by the attack).

To that end, consider again the procedure of the judge: when a customer A complains about a set of n transfers, and the judge thus subpoenas the bank to produce n signed *transfer* forms with distinct numbers N_B, then we have two cases. First, if B does not comply with this subpoena, then B must indeed have broken the law: they were obliged by § 6 only to perform transfers for which they have a corresponding signed *transfer* form, which must all have distinct numbers N_B according to § 6 as each N_B can only be used once. Finally, by § 6, B is also obliged to store each signed *transfer* form and produce them upon a subpoena.[9] Thus, if B does not answer the subpoena correctly, B is rightfully convicted.

Second case, if the bank does correctly answer the subpoena by producing n matching signed *transfer* forms $sign(inv(pk(idp)),T_1),\dots,sign(inv(pk(idp)),T_n)$ with distinct numbers for the N_B-field in the T_i. Then we have to show that the customer is now rightfully convicted. (This is the case where the first flawed version could potentially lead to a wrong conviction.) Recall that we have assumed that the *idp* is honest.[10] This implies that the *idp* has not leaked their private key $inv(pk(idp))$ and thus the signed transfer requests $sign(inv(pk(idp)),T_i)$ have been made by *idp* where, again by honesty of *idp*, we can conclude that the *idp* has followed § 5 when they they signed the transfer forms, i.e., they must have received the $requestpw(A,pw(A),T_i)$ first. Again since the *idp* is honest, one of two things must be the case. First possibility: A has produced all these requests, then A is punishable by § 7 for issuing n request and claiming that they did not. Second possibility: A has leaked $pw(A)$ to somebody else who then produced some of these requests (maybe unbeknownst to A); then A is punishable according to § 3. (Of course, if *idp* were dishonest, it could have leaked $pw(A)$, but we assumed *idp* to be honest.) Thus either way, A is rightfully convicted also in this case and we have established that we are in a lawful system.

9 One may ask whether it is impractical that a bank can always check that the N_B in a request is different from every N_B they have ever accepted, as this seems to imply that the bank always have to consult their entire transaction history. However, following the schema in Fig. 1 is a simple solution: they create a *fresh* random number N_B (that is with overwhelming probability different from all previous such random numbers). The bank remembers N_B *for this session* and accept at most one incoming signed transfer with this N_B. If this does not arrive within a certain time window, the bank simply closes the session and forgets N_B. This is legal, since the bank is not obliged to eventually perform the transfer.

10 This is not an unproblematic assumption as we discuss below, but if we think of a national identity provider, it is at least reasonable that a judge would value their statements (and thus signatures) as trustworthy.

4.2 The Security Goals

The security goals we would like to ensure in this system are:

1. A bank never performs a transfer more often than the customer has ordered it. Observe that this formulation also includes the case that the bank performed a transfer that the customer did not order at all.
2. A customer never complains about a set of bank transfers that they have ordered at least as many times.

The Three Branches of Accountability approach defines a *perfect crime* as an illegal action of an agent where the agent knows they will never be convicted for this action. For instance, in our example, the customer may illegally reuse the same nonce N_A for more than one transaction. As nothing in the described system triggers on that, the agent will never be convicted for that. However, the agent also does not have any advantage from this crime—it is irrelevant.

It is now easy to see that under the assumption that agents will only commit perfect crimes, the security goals hold. Suppose we could reach a state where the first goal is violated, i.e., where a bank has performed a transfer more often than the customer ordered it. Then the client will complain with the judge, which will convict either the client or the bank.[11] Either way, since we have shown the system is lawful, the convicted party has indeed broken the law. The fact that they have been convicted for it shows that it was not a perfect crime, contradicting the assumption. Suppose we could reach a state where the second goal is violated, i.e., a customer dishonestly complains about a set of transfers they have indeed ordered. Again the procedure of the judge will lead to the rightful conviction of either the customer or the bank, which by the perfect crime assumption is absurd. Thus, no state violating the security goals is reachable.

5 Conclusions

Essentially, this paper shows how transactions, like a bank transfer, can be made accountable: at the core the bank needs a transferable proof that the customer indeed ordered the transfer in question. For such a transferable proof, a pure authentication of the customer to the bank is not sufficient, because this would give the bank nothing to convince a third party like a judge. The only option is that we have a signature that could either be produced by the customer themselves or via a trusted third party as in our case study. This solution has the disadvantage that in every transaction, the trusted third party has to be involved, and if it should be hacked for instance, the security guarantees of the entire system are void (as it crucially needs to be an honest party). On the other hand it has the advantage that it can be deployed based on existing identity management infrastructures such as the Danish MitID.

11 Observe here that the client might still be punishable here, even though they did not order all the transfers: they may have given out their password and somebody else did order them.

There are several points one can criticize about the system sketched here. First of all, one may wonder if customers can be blamed if their password gets leaked: of course, not all these leaks are because a customer intentionally gave it to somebody else (like their spouse) but they may have been observed entering it or their computer or phone may have been hacked. This is in fact an old problem that we have for instance for leaked PINs for credit cards: to our knowledge, when a thief withdraws money from an ATM using a stolen card with the correct PIN, the default assumption (unless there is contrary evidence) of the courts is that the card owner must have been careless with their PIN (e.g., kept it written down along with the card). We believe that this is a general dilemma as part of digitalization namely that digital secrets (like PINs, passwords, private keys) have legal meaning, and that losing them can have substantial legal and financial consequences for an individual. There are several mitigations such as multi-factor authentication that at least make it harder for an attack to take over one's identity. One of the anonymous reviewers of this paper suggested that a customer who became victim to a cyber attack could inform the police, and use the police report to get (at least temporarily) reimbursed by the bank. In fact, it is common that the total transfer amounts per day are limited and that banks are ensured against the limited fraudulent transfer that can occur within that time window.

Note, however, that this is a more general problem of digitalization. For our example, suppose the bank loses their database of transfers due to a cyberattack (or simply a software bug) and suddenly become liable in court for all transfers they have executed and cannot justify if subpoenaed for it. This indeed shows that the systems like the one described here cannot be absolute: we have to have human judges who evaluate other evidences like a forensic investigation of a customer's phone for instance. The accountability proof thus provides a strong argument that the system is reasonably well-designed to deter criminals, but it is not an absolute that cannot be overridden when new evidence appears.

In fact, the absolute trust in the *idp* is a second serious problem of this system. A standard approach to replace the single trusted *idp* by a consortium of identity providers of which only a majority needs to be honest is not very promising as this requires a consensus amongst the consortium and is thus not the light-weight solution we have sketched here.

But let us end this list on a third problem for which there is actually a good solution: suppose a customer has obtained via the *idp* a valid signed *transfer* form and send it to the bank. If the bank is malicious, they might hold it off, not executing the transfer but they have the right do so at any moment. The customer is thus in limbo: the transaction has not gone through but could be accepted at any moment. This is a fair exchange problem that in general also requires a trusted third party to resolve; however, one can make this *optimistic* in the sense that the trusted third party would only need to be involved if customer and bank do not come to a consensus on their own [ASW00].

There are several research articles on accountability, most importantly Küsters et al. [KTV10], Künnemann et al. [KGB21], as well as Mödersheim and Cuellar [MC22] that we have used. While the former two are based on computational adversaries, we follow the third approach and consider a Dolev Yao-style intruder who cannot break the cryptography. We have chosen the approach of [MC22] because the concept of the legal system offers the flexibility to support a wide range of systems rather than fixing a particular protocol (like TLS) that is largely irrelevant to the accountability question.

Many works rather use the term *non-repudiation* instead of *accountability*; for a more detailed overview, see for instance [KTV10, MC22]. While the term *non-repudiation*, puts the emphasis on ensuring that actors cannot deny actions they have performed, *accountability* goes further: a bank who has (undeniably) performed a transfer may also be required by a subpoena from a judge to *justify* their action, showing that they acted legally. Depending on the protocol, the answer from an honest actor to a subpoena may give the judge further evidence to investigate [MC22].

As future work, we plan to follow the idea of Bruni et al. [BGS17] to investigate if and how protocol verification tools like Tamarin [Me13], ProVerif [Bl01], or PSPSP [He21] could be employed, and possibly adapted, to verify accountability questions in such an open scenario as the legal system in this paper.

Bibliography

[ASW00] Asokan, N.; Shoup, Victor; Waidner, Michael: Optimistic fair exchange of digital signatures. IEEE J. Sel. Areas Commun., 18(4):593–610, 2000.

[BGS17] Bruni, Alessandro; Giustolisi, Rosario; Schürmann, Carsten: Automated Analysis of Accountability. In (Nguyen, Phong Q.; Zhou, Jianying, eds): ISC 2017. volume 10599. Springer, pp. 417–434, 2017.

[Bl01] Blanchet, Bruno: An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In: Computer Security Foundations Workshop. pp. 82–96, 2001.

[He21] Hess, Andreas V.; Mödersheim, Sebastian; Brucker, Achim D.; Schlichtkrull, Anders: Performing Security Proofs of Stateful Protocols. In: 34th IEEE Computer Security Foundations Symposium (CSF). volume 1. IEEE, pp. 143–158, 2021.

[KGB21] Künnemann, Robert; Garg, Deepak; Backes, Michael: Accountability in the Decentralised-Adversary Setting. In: 2021 IEEE 34th Computer Security Foundations Symposium (CSF). IEEE Computer Society, pp. 95–110, 2021.

[KTV10] Küsters, Ralf; Truderung, Tomasz; Vogt, Andreas: Accountability: definition and relationship to verifiability. In: Proceedings of the 17th ACM conference on Computer and Communications Security. pp. 526–535, 2010.

[MC22] Mödersheim, Sebastian; Cuellar, Jorge: Three Branches of Accountability. In: In Festschrift for Joshua Guttman, LNCS 13066, 2021. . Springer, 2022.

[Me13] Meier, Simon; Schmidt, Benedikt; Cremers, Cas; Basin, David A.: The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In: Computer Aided Verification. pp. 696–701, 2013.