# Alternative Tabellenschutzfunktionen

Autoren: Wolfram E.Mewes, Bernd Held

**Dieser Artikel beschreibt alternative Möglichkeiten, wie Sie Ihre Daten in Excel schützen können. Dabei werden sowohl programmierte Lösungen als auch zwei Standard-Lösungen ohne VBA präsentiert.**

Der Tabellenschutz über das Menü *Extras* und dem Befehl *Schutz* stellt die Standardmöglichkeit in Excel dar, um Tabellen sowie Zellen vor Veränderungen zu schützen. Zu diesem Zweck kann ein Passwort eingegeben werden. Die Passworteingabe ist zwar nicht zwingend notwendig, aber es empfiehlt sich trotzdem, bei diesem Vorgang das Passwort anzugeben. Nur so ist „sichergestellt", dass man die Tabelle später wieder für eigene Änderung „entschützen" kann, ohne befürchten zu müssen, dass andere Anwender die Tabelle in der Zwischenzeit verändert haben. Im Internet werden jedoch auf diversen Homepages privater und auch geschäftlicher Art sogenannte Passwort-Knacker angeboten, sodass der Tabellenschutz heutzutage keinen wirkungsvollen Schutz mehr darstellt. Ein weiterer Punkt ist, dass man selbst als Anwender nicht davor geschützt ist, seine eigenen Passwörter zu vergessen. Ein langer Urlaub oder auch zuviel Streß lassen dabei schnell einmal das ein oder andere Passwort vergessen. Es ist daher Zeit, ein paar alternative Ideen zu entwickeln, die man stattdessen beziehungsweise als Ergänzung einsetzen kann, um wertvolle Daten vor Veränderungen zu schützen. Dieser Artikel stellt einige Verfahren vor, mithilfe derer Sie eine zusätzliche Sicherung in Ihre Mappen einbauen können, ohne auf das Standardverfahren zuzugreifen. Einen wirklichen hundertprozentigen Schutz wird es aber mit Sicherheit für Excel-Tabellen nie geben. Trotzdem kann man es „Hackern" so schwer wie möglich mit den folgenden Verfahren machen.

**Zugang per Doppelklick**

In der ersten Variante, die Sie in der Arbeitsmappe *DatenSchützen.xls* auf der CDROM einsehen können, wird schon beim Öffnen der Arbeitsmappe dafür gesorgt, dass alle Tabellen, bis auf die erste Tabelle, ausgeblendet werden. Außerdem wird die erste Tabelle der Arbeitsmappe aktiviert. Als kleine Vorarbeit legt man auf der ersten Tabelle einen Bereich von Zelle C2:F10 an, weist diesen Zellen einen Rahmen sowie eine Hintergrundfarbe zu und richtet die Spaltenbreite und Zeilenhöhe so ein, dass der Eindruck entsteht, es handle sich um „Schaltflächen" (siehe Bild 1) auf einem Eingabe-Panel. Innerhalb von diesem Bereich sollen später drei Felder (=Zellen) nacheinander doppelt angeklickt werden. Die so doppelt angeklickten Felder werden danach sofort mit einer anderen Hintergrundfarbe (Rot) formatiert. Nur wenn die richtigen Felder und die korrekte Reihenfolge (im Beispiel D4, E6, F4) der drei Felder eingehalten wird, dann werden die vorher ausgeblendeten Tabellen wieder verfügbar gemacht.

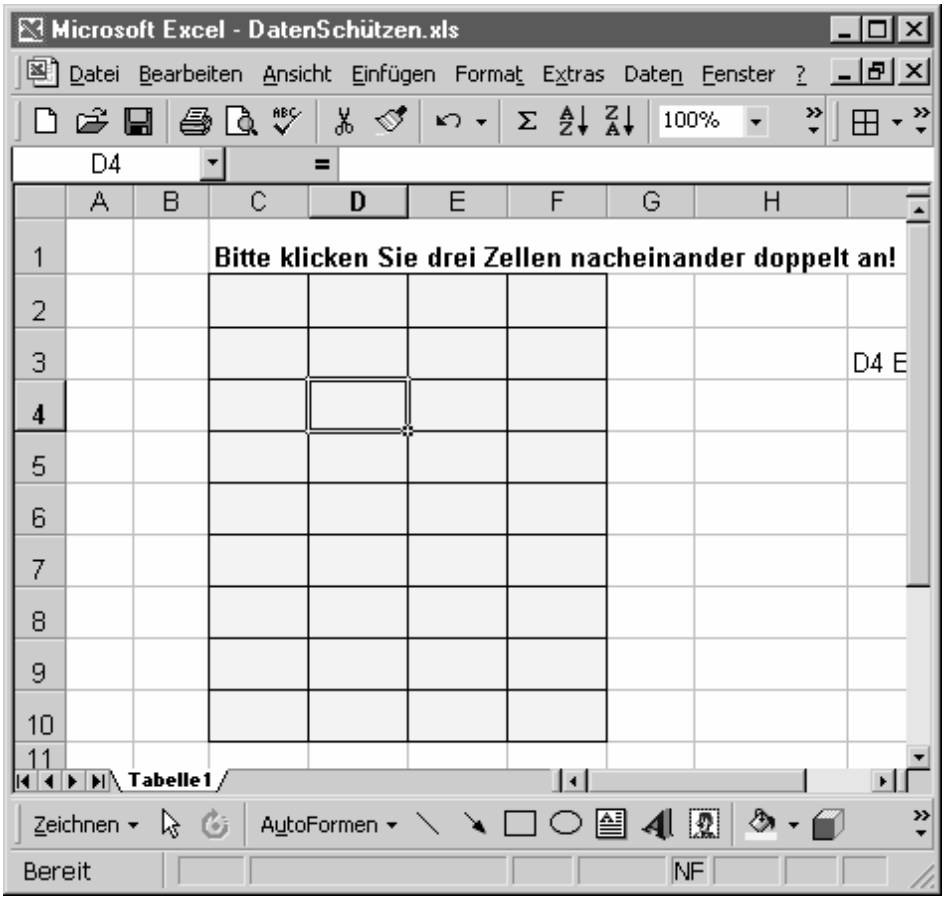

Bild 1: Die Ausgangstabelle.

Nachdem diese Vorarbeit geleistet wurde, erstellt man die Ereignisprozedur *Workbook_Open*, die automatisch beim Öffnen der Arbeitsmappe ausgeführt wird. Um diese Aufgabe umzusetzen, wird wie folgt vorgegangen:

1. Über die Tastenkombination <Alt> + <F11> wird in die Entwicklungsumgebung von Excel verzweigt.
2. Im Projekt-Explorer erfolgt ein Doppelklick auf den Eintrag *DieseArbeitsmappe*.
3. Dann wird folgendes Ereignismakro aus Listing 1 erfasst:

Listing 1 (Arbeiten, die beim Öffnen der Arbeitsmappe ausgeführt werden)

```
Private Sub Workbook_Open()
Dim TabZähler As Integer
Dim Zielbereich As Range

Sheets("Tabelle1").Activate
Set Zielbereich = Sheets("Tabelle1").Range("C2:F10")

For TabZähler = 2 To ActiveWorkbook.Worksheets.Count
 Sheets(TabZähler).Visible = xlVeryHidden
Next TabZähler

'Ausgangsfarbe herstellen
For Each zelle In Zielbereich
 zelle.Interior.ColorIndex = 36
Next zelle
End Sub
```

Im ersten Schritt des Makro aus Listing 1 wird eine Variable names *TabZähler* vom Datentyp *Integer* erstellt. Mit dem Inhalt dieser Variablen wird später die Schleifensteuerung vorgenommen. Als zweite Variable benötigt man eine Variable vom Typ *Range*. Dort soll später der Zielbereich C2:F10 verwaltet und überwacht werden. Danach wird die *Tabelle1* mithilfe der Methode *Activate* angesprungen. Direkt im Anschluss daran weist man der Variablen *Zielbereich* den Bereich der Tabelle zu, in dem nachher per Doppelklick die „Eingaben“ vorgenommen werden sollen.

In der nachfolgenden Schleife wird beginnend von der zweiten Tabelle der Arbeitsmappe dafür gesorgt, dass alle Tabellenblätter über die Eigenschaft *Visible* ausgeblendet werden. Wieviele Tabellenblätter sich in der Arbeitsmappe befinden, kann man über die Anweisung *Activeworkbook.Worksheets.Count* ermitteln. Bei der Eigenschaft *Visible* ist hinzuzufügen, dass die Konstante *xlVeryHidden* dafür

sorgt, dass die so ausgeblendeten Tabellen nicht mehr über den Menübefehl *Format/Blatt/Einblenden* eingeblendet werden können. Man spricht bei diesem Verfahren auch von „Sicherem Ausblenden“. Ein Einblenden dieser Tabellen ist demnach nur über ein Makro beziehungsweise über den Umweg über die Entwicklungsumgebung (siehe Ende des Artikels) möglich. Selbstverständlich ist es auch möglich, hier der Eigenschaft *Visible* den Wert *False* zuzuweisen. In diesem Fall aber, ist es möglich, die ausgeblendeten Tabellen über die Standardoberfläche von Excel also über den Menübefehl *Format/Blatt/Einblenden* wieder verfügbar zu machen.
In der nächsten Schleife wird dafür gesorgt, dass eine einheitliche Formatierung aller Zellen des Zielbereichs vorgenommen wird. Später sollen schließlich die doppelt angeklickten Zellen anders gefärbt werden. Mit dieser Schleife stellen Sie demnach den Ausgangszustand (siehe Bild 1) wieder her.
Die eigentliche Überprüfung der doppelt angeklickten Felder erfolgt direkt hinter der *Tabelle1*. Dazu klickt man mit der rechten Maustaste auf die Registerkarte dieser Tabelle und wählt den Befehl *Code anzeigen* aus dem Kontextmenü. Danach wird das Ereignis *Worksheet_BeforeDoubleClick* aus Listing 2 erfasst, das automatisch ausgeführt wird, wenn ein Doppelklick auf eine Zelle durchgeführt wird.
Listing 2 (Die Doppelklicks werden abgefangen und ausgewertet)

```
Dim ZellenAdressen As String
Dim Klickzähler As Integer

Private Sub Worksheet_BeforeDoubleClick _
(ByVal Target As Range, Cancel As Boolean)
Dim Zielbereich As Range
Dim TabZähler As Integer
Dim zelle As Range

Set Zielbereich = Sheets("Tabelle1").Range("C2:F10")
If Intersect(Target, Zielbereich) Is Nothing Then Exit Sub
Cancel = True
ZellenAdressen = ZellenAdressen & Target.Address
Klickzähler = Klickzähler + 1
Target.Interior.ColorIndex = 3

If Klickzähler = 3 Then
 Klickzähler = 0
 If ZellenAdressen = "$D$4$E$6$F$4" Then
  MsgBox "Zugang richtig!", vbInformation
  For TabZähler = 2 To ActiveWorkbook.Worksheets.Count
   Worksheets(TabZähler).Visible = True
  Next TabZähler
  Else
   MsgBox "Zugang wird verweigert!", vbCritical
   ZellenAdressen = ""
   Klickzähler = 0
   For Each zelle In Zielbereich
       zelle.Interior.ColorIndex = 36
   Next zelle
  End If
Else
End If
End Sub
```

Deklariert werden im ersten Schritt zwei globale Variablen, die vor der eigentlichen Ereignisprozedur angelegt werden. Über die Variable *ZellenAdressen* werden die Zellenkoordinaten der doppelt angeklickten Zellen gespeichert, über die Variable *KlickZähler* wird die Anzahl der durchgeführten Doppelklicks gezählt und verwaltet. Innerhalb der Ereignisprozedur *Worksheet_BeforeDounbleClick* wird die Variable *Zielbereich* vom Datentyp *Range* deklariert. In dieser Variablen wird kurz danach der Zielbereich C2:F10 über die Anweisung *Set* angegeben. Die Variable *TabZähler* wird dazu eingesetzt, um die Anzahl der in der Arbeitsmappe enthaltenen Tabellen über die Anweisung *Activeworkbook.Worksheets.Count* zu ermitteln. Die Variable *zelle* wird am Ende des Makros benötigt, um alle Zellen im Zielbereich farblich über die Eigenschaft *ColorIndex* auf den Ausgangszustand zurückzusetzen.
In der ersten Schleife des Makros aus Listing 2 wird geprüft, ob der Doppelklick innerhalb des Zielbereichs stattfand. Dazu verwendet man die Anweisung *If Intersect(Target, Zielbereich) Is Nothing Then Exit Sub*. Erfolgt ein Doppelklick außerhalb des Zielbereichs, dann wird über die Anweisung *Exit Sub* direkt aus der Ereignisprozedur gesprungen. Im anderen Fall wird die Verarbeitung fortgesetzt. Das Argument *Cancel* wird als Nächstes auf den Wert *True* gesetzt. Damit wird die standardmäßig vorgesehene Aktion (die direkte Zellenbearbeitung) verhindert. In der Variablen *ZellenAdressen* wird jetzt über die Anweisung *ZellenAdressen = ZellenAdressen & Target.Address* die jeweils doppelt angeklickten Zellenadressen zusammengeführt sowie der *KlickZähler* um den Wert 1 erhöht. Außerdem wird

die so angeklickte Zelle mit der Hintergrundfarbe *Rot* formatiert, indem der jeweiligen Zelle über die Eigenschaft *ColorIndex* der Wert 3 zugewiesen wird.

Im folgenden Schritt muss geprüft werden, ob bereits drei Doppelklicks auf Zellen durchgeführt wurden. Wenn ja, dann wird der *KlickZähler* sicherheitshalber auf den Wert 0 gesetzt. Im Anschluß daran wird die Variable *ZellenAdressen* mit dem String "$D$4$E$6$F$4" verglichen. Ist dieser Vergleich erfolgreich, dann werden über eine Schleife die versteckten Tabellenblätter nacheinander eingeblendet. Im anderen Fall, wird eine Fehlermeldung am Bildschirm gezeigt, die Ausgangsformatierung der Tabelle wiederhergestellt sowie die Variablen *KlickZähler* und *Zellenadressen* initialisiert.

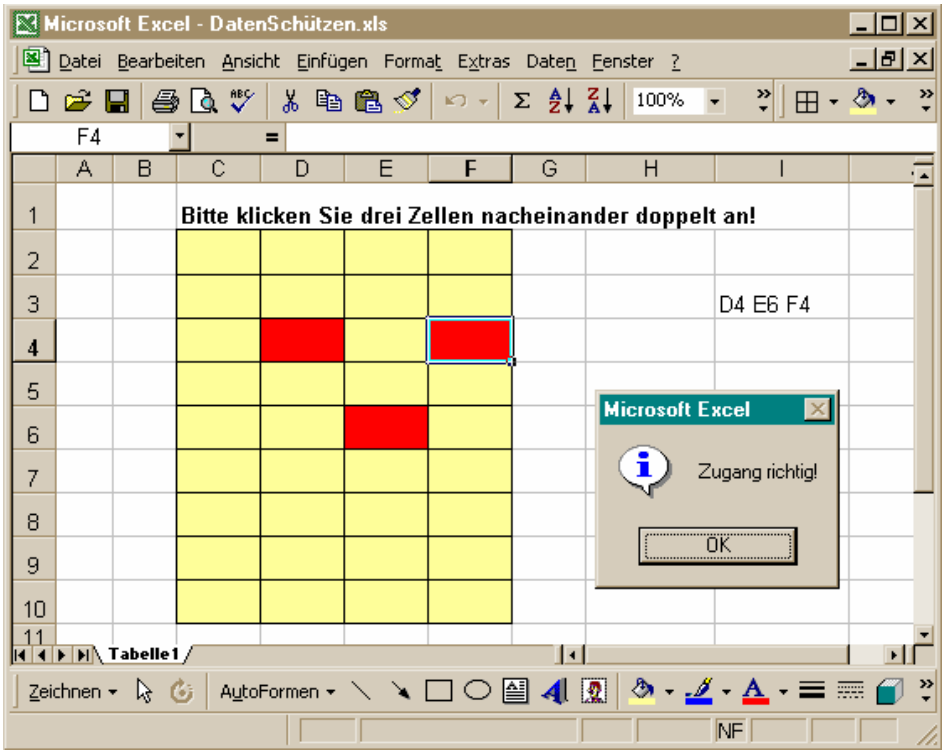

Bild 2: Die Zellen wurden in der richtigen Reihenfolge doppelt angeklickt.

**Makro-Lösung schützen**

Da diese Lösung über ein Makros realisiert wurde und die Reihenfolge der doppelt angeklickten Zellen als Text „$D$4$E$6$F$4" direkt aus dem Code ablesbar ist, sollte man den Zugang zu den Makros sperren. Dazu wechselt man in die Entwicklungsumgebung und führt im Projekt-Explorer mit der rechten Maustaste einen Klick auf die Arbeitsmappe durch und wählt den Befehl *Eigenschaften von VBA-Project*. Im Dialog *VBA-Project - Projekteigenschaften* wechselt man auf die Registerkarte *Schutz*, aktiviert das Kontrollkästchen *Projekt für die Anzeige sperren*, vergibt und bestätigt ein Kennwort und endet die Aktion mit *OK* ab. Der Schutz wird jedoch erst wirksam, wenn die Arbeitsmappe gespeichert, geschlossen und erneut wieder geöffnet wird.

**Zugriffssteuerung in Abhängigkeit vom Anwender**

In der nächsten Variante wird eine Arbeitsmappe (*DatenSchützen2.xls*) so gesichert, dass schon beim Öffnen der Arbeitsmappe geprüft wird, wer am PC angemeldet ist. Nur wenn ein bestimmter Anwender (im Beispiel ist das der Anwender *Held*) angemeldet ist, dann erhält der Zugreifer Zugriff auf die gesamte *Tabelle1*. Im anderen Fall wird eine ScrollArea im Zellenbereich A1:F15 eingerichtet. Eine ScrollArea bedeutet, dass ein Anwender sich nur in einem ganz bestimmten Zellenbereich aufhalten darf. Der Anwender kann auf einer Tabelle, auf der eine ScrollArea eingestellt ist, keine andere Zelle außerhalb dieses Bereichs anspringen und schon gar nicht editieren.

Um diese Aufgabenstellung zu lösen, geht man wie folgt vor:

1. Über die Tastenkombination <Alt> + <F11> wird in die Entwicklungsumgebung von Excel verzweigt.
2. Im Projekt-Explorer erfolgt ein Doppelklick auf den Eintrag *DieseArbeitsmappe*.
3. Dann wird folgendes Makro aus Listing 2 erfasst:

Listing 3 (Beim Öffnen der Arbeitsmappe eine Funktion aufrufen)

```
Private Sub Workbook_Open()
Select Case Anwender
Case "Held"
 'ScrollArea aufheben
 Sheets("Tabelle1").ScrollArea = ""
Case Else
'ScrollArea einrichten
MsgBox "Sie sind nicht der erwartete Anwender! " & _
" Eine ScrollArea wird eingerichtet", vbCritical
 Sheets("Tabelle1").ScrollArea = "A1:F15"
End Select
End Sub
```

In einer *Select Case* Auswertung wird zunächst eine Funktion namens *Anwender* aufgerufen. In Abhängigkeit vom Rückgabewert dieser Funktion wird die ScrollArea eingerichtet bzw. gelöscht. Was momentan noch fehlt ist die Funktion *Anwender*, die man in einem neuen Standardmodul *Modul1* erfasst. Sehen Sie sich dazu das Listing 4 an.

Listing 4 (Prüfen, ob der richtige Anwender vor dem PC sitzt)

```
Private Declare Function GetUserName Lib "advapi32.dll" _
   Alias "GetUserNameA" _
  (ByVal lpBuffer As String, nSize As Long) As Long

Public Function Anwender() As String
Dim Namen As String

   Namen = Space$(256)

   If GetUserName(Namen, Len(Namen)) Then
     Anwender = Left(Namen, InStr(Namen, Chr$(0)) - 1)
   End If
End Function
```

Die Funktion *Anwender* ruft wiederum eine Funkton auf. Bei der aufgerufenen Funktion *GetUserName* handelt es sich um eine sogeanannte API-Funktion, die standardmäßig mit dem Betriebssystem Windows ausgeliefert wird. Solche API-Funktionen kann man übrigens in nahezu allen Programmiersprachen einsetzen. Diese Funktion liefert den Anwendernamen zurück, der in Windows momentan angemeldet ist. Sollten in diesem Namen Leerzeichen am Ende enthalten sein, dann sorgt die Funktion *Left* dafür, dass diese am Ende des Strings entfernt werden.


Bild 3: Die Prüfung für den Zugang erfolgt auf Anwendernamen.

**Zugriffssteuerung in Abhängigkeit vom PC**

Die dritte Variante sorgt dafür, dass eine Arbeitsmappe nur dann in vollem Umfang genutzt werden darf, sofern die Mappe auf einem bestimmten PC geöffnet wird. Im folgenden Beispiel (*DatenSchützen3.xls*) wird beim Öffnen der Arbeitsmappe geprüft, ob die Mappe von einem bestimmten PC (im Beispiel ist das der PC mit dem Namen HELD-OFFICE 1) aus geöffnet wurde. Auch in diesem Fall setzt man das Ereignis *Workbook_Open* ein, um die Prüfung direkt beim Öffnen der Arbeitsmappe durchzuführen. Sollte die Mappe auf einem anderen PC geöffnet werden, wird die Mappe augenblicklich wieder geschlossen, ohne dass Änderungen gespeichert werden. In der Entwicklungsumgebung wird daher der Code aus Listing 5 hinter dem Eintrag *DieseArbeitsmappe* eingegeben.

Listing 5 (Die Funktion PC wird aufgerufen)

```
Private Sub Workbook_Open()
Select Case PC
Case "HELD-OFFICE 1"
 'OK- keine weiteren Aktionen
Case Else
 MsgBox "Sie sind nicht der erwartete PC! " & _
" Die Mappe wird wieder geschlossen", vbCritical
 ActiveWorkbook.Close savechanges:=False
End Select
End Sub
```

**Hinweis:** Um das in der Demodatei eingestellte Ereignis beim PC-Namen anzupassen, öffnen Sie die Arbeitsmappe *DatenSchützen3.xls*, indem Sie die Taste <Umschalt> beim Öffnenvorgang gedrückt halten. Somit wird das Ereignis *Workbook_Open* nicht ausgeführt und Sie können den Code anpassen, speichern, schließen und wieder öffnen. Übrigens kann das *Workbook_Open* Ereignis als einzigstes Ereignis direkt aus der Entwicklungsumgebung heraus gestartet werden. Es wird also nicht unbedingt mehr nötig, die Arbeitsmappe nach dem Anpassen erneut zu öffnen.

Innerhalb des Ereignisses *Workbook_Open* wird die Funktion *PC* aufgerufen. Diese Funktion wird in einem Standardmodul (*Modul1*) erfasst, welches Sie in Listing 6 sehen können.

Listing 6 (Prüfen, ob die Mappe vom richtigen PC aus geöffnet wurde)

```
  Private Declare Function GetComputerName Lib "kernel32" _
     Alias "GetComputerNameA" _
    (ByVal lpBuffer As String, nSize As Long) As Long

  Public Function PC() As String
  Dim Namen As String

     Namen = Space$(256)

     If GetComputerName(Namen, Len(Namen)) Then
       PC = Left(Namen, InStr(Namen, Chr$(0)) - 1)
     End If
  End Function
```

**Hinweis:** Über das Schlüsselwort *Public* wird die Funktion als öffentlich deklariert, was bedeutet, dass Sie diese Funktion auch aus anderen Modulen aufrufen können.

Die API-Funktion *GetComputerName* liefert den Namen des PCs wie er in der Systemsteuerung von Windows hinterlegt ist.

**Eingaben sofort wieder verschwinden lassen**

Eine weitere Variante, bei der VBA zum Einsatz kommt, ist die, dass Eingaben in bestimmten Bereichen sofort wieder rückgängig gemacht werden. Zu diesem Zweck legen Sie eine neue, noch leere Arbeitsmappe mit einer Tabelle an und speichern diese unter dem Namen *DatenSchützen4.xls*. Formatieren Sie danach den Bereich B3:E10 mit der Hintergrundfarbe Hellgelb und geben einige Zahlenwerte, wie beispielsweise in Bild 4 angezeigt, ein.

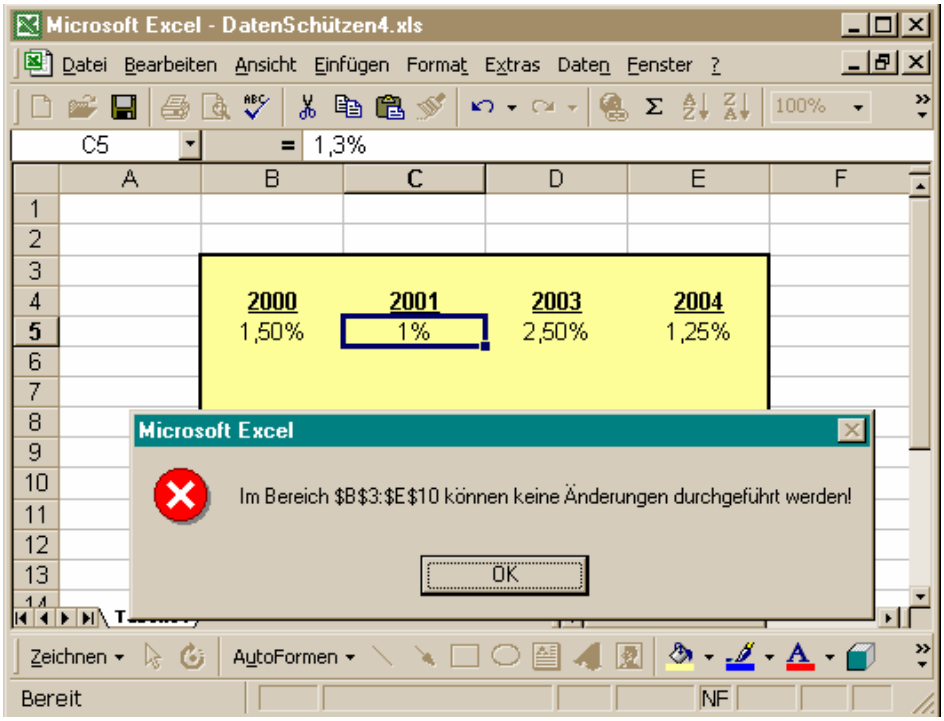

Bild 4: Diese Daten sollen geschützt werden.

Die Aufgabe besteht nun darin, die Daten im Bereich B3:E10 zu schüzen, ohne dabei ein Kennwort zu verwenden. Um diese Aufgabe zu lösen, befolgen Sie die nächsten Arbeitsschritte:

1. Klicken Sie mit der rechten Maustaste auf den Tabellenreiter der *Tabelle1* und wählen den Befehl Code anzeigen aus dem Kontextmenü.
2. Wählen Sie in der Entwicklungsumgebung im Codebereich aus dem ersten Dropdown den Befehl *Worksheet*.
3. Aus dem zweiten Dropdown wählen Sie den Befehl *Change*.
4. Ergänzen Sie den leeren Ereignisrahmen um die folgenden Zeilen aus Listing 7.

Listing 7 (Eingaben im Zielbereich werden sofort widerrufen)

```
Private Sub Worksheet_Change(ByVal Target As Range)
Dim Zielbereich As Range

Set Zielbereich = Sheets("Tabelle1").Range("B3:E10")
If Intersect(Target, Zielbereich) Is Nothing Then Exit Sub
Application.EnableEvents = False
MsgBox "Im Bereich " & Zielbereich.Address & vbLf & _
" können keine Änderungen durchgeführt werden!", vbCritical
Application.Undo
Application.EnableEvents = True
End Sub
```

Deklarieren Sie im ersten Schritt die Variable *Zielbereich* vom Typ *Range* und weisen dieser Variablen mithilfe der Anweisung *Set* den Zellenbereich B3:E10 zu. Überprüfen Sie anschließend mithilfe der Methode *Intersect*, ob die Eingabe im Zielbereich vorgenommen wurde. Wenn ja, dann schalten Sie zunächst die Ereignissteuerung über die Anweisung *Application.EnableEvents=False* aus und widerrufen die Eingabe mithilfe der Anweisung *Application.Undo*. Jetzt schalten Sie die Ereignissteuerung wieder ein, indem Sie der Eigenschaft *Enabled* den Wert *True* zuweisen. Dieses Ein- und Ausschalten der Ereignissteuerung von Excel ist zwingend notwendig, damit keine Endlosschleife produziert wird, die automatisch auftreten würde, wenn Sie den Vorgang der Eingabe rückgängig machen.

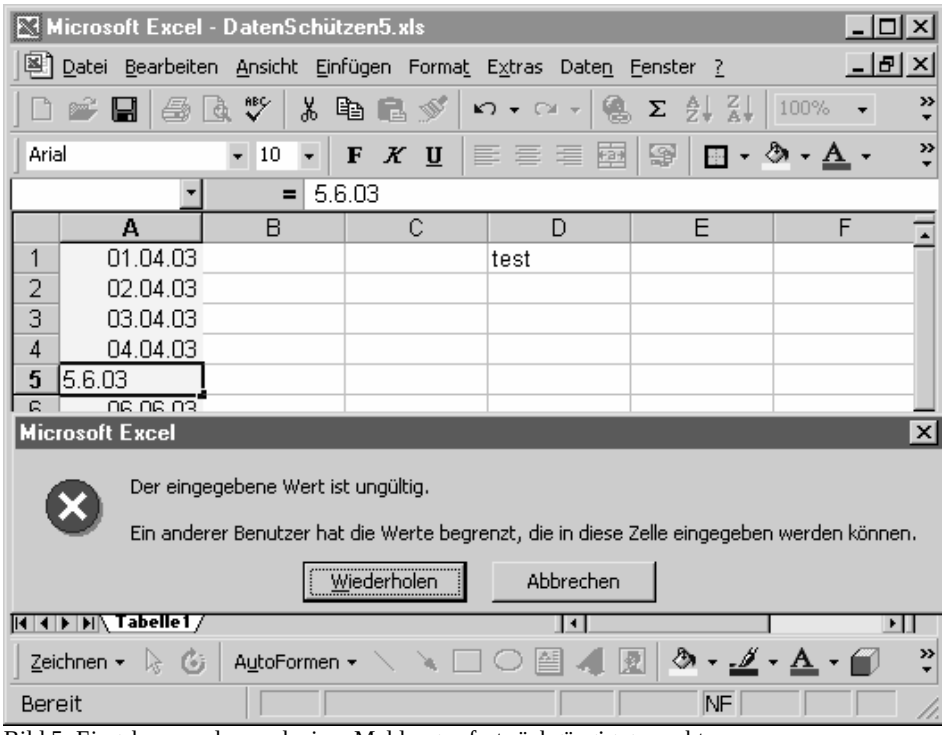

Bild 5: Eingaben werden nach einer Meldung sofort rückgängig gemacht.

**Hinweis:** Wenn Sie temporär Änderungen im Bereich wieder zulassen möchten, dann müssen Sie das Ereignis *Worksheet_Change* kurzfristig auskommentieren. Dazu wechseln Sie in die Entwicklungsumgebung und blenden die Symbolleiste *Bearbeiten* ein. Markieren Sie danach das komplette Ereignis und klicken in der Symbolleiste *Bearbeiten* auf das Symbol *Block auskommentieren*. Damit wird vor jede Zeile des Ereignismakros ein Apostroph eingefügt und der Quelltext wird automatisch mit der Schriftfarbe *Grün* formatiert. Das Ereignismakro ist somit deaktiviert.

**Eingaben verhindern über die Gültigkeitsfunktion**

Als nächste Variante soll eine Schutzmaßnahme vorgestellt werden, die ohne den Einsatz von Programmierung auskommt. Legen Sie dazu eine neue Arbeitsmappe mit einer Tabelle an und speichern diese unter dem Namen *DatenSchützen5.xls*. Schreiben Sie in Zelle A1 das heutige Datum und in Zelle A2 die Formel =A1+1. Kopieren Sie diese Formel soweit Sie möchten nach unten. In *Tabelle1* soll diese Datumsspalte nun komplett vor Eingaben geschützt werden.

Schreiben Sie als Nächstes den Text OK in Zelle D1. Befolgen Sie danach die nächsten Arbeitsschritte:

1. Markieren Sie alle gefüllten Zellen der Spalte A.
2. Wählen Sie aus dem Menü *Daten* den Befehl *Gültigkeit*.
3. Im Dialog *Gültigkeitsprüfung* wechseln Sie auf die Registerkarte *Einstellungen*.
4. Wählen Sie aus dem Dropdown *Zulassen* den Befehl *Benutzerdefiniert*.
5. Geben Sie im Feld *Formel* folgende Formel ein: =$D$1="OK"
6. Wechseln Sie auf die Registerkarte *Fehlermeldung* und stellen Sie sicher, dass hier das Symbol *Stopp* eingestellt ist.
7. Bestätigen Sie diese Einstellung mit *OK*.

In der Spalte A sind Änderungen jetzt nur möglich, wenn in Zelle D1 der Text *OK* steht. Ändern Sie testweise diesen Wert aus Zelle D1 in einen anderen Wert und versuchen Sie dann ein neues Datum in Spalte A einzugeben. Excel wird dies verweigern!

**Tipp:**

Um den Eintrag in Zelle D1 (OK) nicht gleich für jedermann ersichtlich zu machen, können Sie die Zelle mit dem benutzerdefinierten Format ;;; (dreimal das Semikolon) formatieren. Eine alternative Möglichkeit wäre die Zelle mit der Schriftfarbe *Weiß* zu formatieren, damit der Text sich nicht mehr vom Hintergrund

abhebt. Zusätzlich können Sie die komplette Spalte D ausblenden, um diese „Schlüsselzelle“ zu verbergen.

**Tabellen sicher ausblenden**

Über den Menübefehl *Format/Blatt ausblenden* können Sie Tabellen verbergen, die nicht auf den ersten Anblick sichtbar sein sollen. Die so versteckten Tabellen können aber jederzeit wieder über den Menübefehl *Format/Blatt/Einblenden* verfügbar gemacht werden. In der ersten Variante dieses Artikels wurde bereits ein sicheres Ausblenden per Makro vorgestellt. Dieses Verfahren können Sie jedoch auch ohne Makro durchführen, indem Sie wie folgt vorgehen:

1. Wechseln Sie über die Tastenkombination <Alt> + <F11> in die Entwicklungsumgebung von Excel.
2. Im Projekt-Explorer klicken Sie die Tabelle an, die Sie sicher ausblenden möchten.
3. Im Eigenschaften-Fenster setzen Sie den Mauszeiger auf das Eigenschaftsfeld *Visible*.

Bild 6: Tabellen sicher verstecken.

4. Wählen Sie im daneben liegenden Feld den Eintrag *2- xLSheetVeryHidden* aus.
5. Betätigen Sie die Taste <Tabulator>.

Die so ausgeblendeten Tabellen können über den Menübefehl *Format/Blatt/Einblenden* nicht mehr eingeblendet werden. Um so ausgeblendete Tabellen wieder einzublenden, muss man den Weg über die Entwicklungsumgebung, den Projekt-Explorer sowie das Eigenschaften-Fenster gehen. Damit haben Sie zwar die Entwicklungsumgebung von Excel genützt, ohne aber ein Makro einzusetzen.